DSTO

Software Function Summaries for the Low Speed Wind Tunnel Data Acquisition System

Craig D. Edwards,
Owen F. Holland,
Stephen S.W. Lam,
Sunny Y.F. Leung, Yoel Y. Link
and Robert Toffoletto

DSTO-TN-0321

20010105 124

Software Function Summaries for the Low Speed Wind Tunnel Data Acquisition System

Craig D. Edwards, Owen F. Holland, Stephen S. W. Lam,
Sunny Y. F. Leung, Yoel Y. Link and
Robert Toffoletto

Air Operations Division
Aeronautical and Maritime Research Laboratory

DSTO-TN-0321

ABSTRACT

The data acquisition system in the Low Speed Wind Tunnel at the Aeronautical and Maritime Research Laboratory was recently upgraded. Several software packages were developed using C, X/Motif, and OpenGL programming languages and libraries. The software provides wind tunnel operators with graphical user interfaces from which test data can be easily monitored, acquired and processed. Descriptions of all software source code functions have been compiled and included as Appendices in this document. These function summaries have also been published on the Defence Science and Technology Organisation Intranet for easy reference for future software development.

RELEASE LIMITATION

Approved for public release

DEPARTMENT OF DEFENCE
DEFENCE SCIENCE & TECHNOLOGY ORGANISATION
DSTO

DTIC QUALITY INSPECTED 4

APPROVED FOR PUBLIC RELEASE

Software Function Summaries for the Low Speed Wind Tunnel Data Acquisition System

Executive Summary

The data acquisition system in the Low Speed Wind Tunnel (LSWT) at the Aeronautical and Maritime Research Laboratory (AMRL) was recently upgraded. The new system consists of a Digital AlphaServer 400 as the host computer with communication to several PC-based and VME-based slave instrumentation modules. VME-based modules were connected serially to a PC and communication to all PC-based modules was achieved using Ethernet standard.

In this environment, several software packages were developed using C, X/Motif, and OpenGL programming languages. This software included the data communication process and several applications to assist the operators in the configuration of wind tunnel tests, monitoring of parameters, and the acquisition and reduction of test data.

Descriptions of all these applications' source-code functions have been written and are documented in this report. This compilation serves as a basis and as a useful reference guide for future software development in the LSWT. The function summaries have also been published on the Defence Science and Technology Organisation (DSTO) Intranet at the following URL location:

http://bernoulli.dsto.defence.gov.au/data_acq/Software/Files/Index.html

Authors
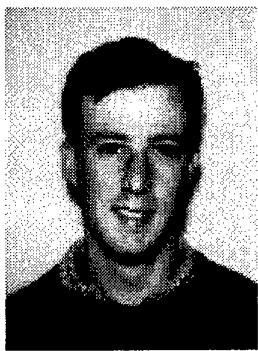
Craig D. Edwards
Air Operations Division

Craig graduated from the University of Queensland in 1995 completing a Bachelor of Mechanical and Space Engineering with First Class Honours. The following year he obtained employment with the Aeronautical and Maritime Research Laboratory at Melbourne. Working in Flight Mechanics, he has gained considerable experience in the area of wind tunnels and experimental aerodynamics. This has included test programmes with the Hydrographic Ship and PC-9/A aircraft in the Low Speed Wind Tunnel, and contributions to the F-111/AGM-142 store clearance project in the Transonic Wind Tunnel. He has also been involved extensively in the development of wind tunnel data acquisition systems.
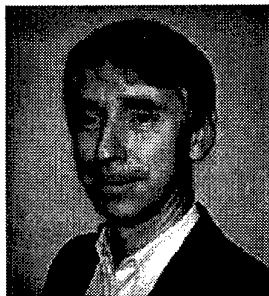
Owen F. Holland
Air Operations Division

Owen Holland graduated from Bendigo College of Advanced Education in 1977 with a Diploma of Electronic Engineering and then obtained a Bachelor of Engineering (Electronic) from Royal Melbourne Institute of Technology in 1983. In 1996 he was awarded a Master of Engineering in Electrical and Electronic Engineering from Victoria University of Technology. He commenced employment at the Aeronautical Research Laboratory in 1978. Since then he has worked in a number of areas including instrumentation and data acquisition systems for wind tunnels. He is currently a Senior Professional Officer in the Air Operations Division at the Aeronautical and Maritime Research Laboratory.
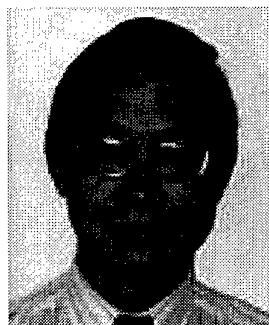
Stephen S. W. Lam
Air Operations Division

Dr Stephen Lam graduated as a Bachelor of Engineer (Mechanical) in 1979 and obtained a degree of Master of Engineering Science in 1982 from the University of Melbourne. He later undertook a research study on Natural Convection in Trapezoidal Cavities at Monash University and was awarded the degree of Doctor of Philosophy in 1990. Dr Lam joined DSTO in 1988 and has since been working in the area of wind tunnel research. He has implemented a machine to calibrate wind tunnel strain gauge balances, and took a leading role in the implementation of a new data acquisition system for both the Low Speed and Transonic Wind Tunnels at AMRL. Dr Lam is now actively involved in the implementation of advanced aerodynamic testing techniques, and is investigating the application of Pressure Sensitive Paint measurement technique in both the transonic and low speed flow regimes.

Sunny Y. F. Leung
Air Operations Division

Sunny Leung completed his Bachelor of Engineering in Aerospace Engeering and Bachelor of Business in Business Administration with First Class Honours at RMIT in 1998, and joined the Aeronautical and Maritime Research Laboratory at Fishermans Bend the following year. He worked in the Flight Mechanics area of the Air Operations Division, and was involved with experimental aerodynamics and wind tunnel test techniques. He also worked extensively in the investigation of strain gauge balance calibration matrix mathematical models.

Yoel Y. Link
Air Operations Division

Yoel Link completed his Bachelor of Science in 1987 and his Bachelor of Engineering in Aeronautical Engineering in 1989, both at Sydney University, and he joined the Aeronautical Research Laboratory at Melbourne the following year. He completed a Master of Business Administration in Technology Management in 1995 at Monash University. He has predominantly worked in Flight Mechanics and experimental aerodynamics in the Wind Tunnels. During this period he has accumulated extensive experience in aerodynamics with the F-111, AGM-142, Jindivik, Tonic, PC-9, Mk82, Amphibious Transport (LPA) ship, and the Hydrographic Ship wind tunnel test programmes. He has also been responsible for the development of the wind tunnel data acquisition systems, and is currently responsible for the new Transonic Wind Tunnel, which was completed in March 2000.

Robert Toffoletto
Air Operations Division

Robert Toffoletto graduated from the Royal Melbourne Institute of Technology in 1986 having obtained an Aeronautical Engineering degree. Since commencing employment at the then Aeronautical Research Laboratory in 1987, he has been involved with the aerodynamic modelling of a number of helicopters using a comprehensive rotorcraft analysis code and has used an aerodynamic panel code to develop a model of the F/A-18 aircraft and the PC-9/A trainer. He has also conducted a number of wind-tunnel tests including an investigation of the boundary layer around an array of hydrophones, flow visualisation about a PC-9/A wind tunnel model, and calibration of a rake of pressure probes.

Contents

1. Introduction

The data acquisition system in the Low Speed Wind Tunnel (LSWT) at the Aeronautical and Maritime Research Laboratory (AMRL) was recently upgraded (Holland *et al*, 1999). The new system consists of a Digital AlphaServer 400 as the host computer which provided a C, X/Motif, and OpenGL software development environment. Communication to PC-based slave instrumentation modules was achieved using 10 Megabit per second Ethernet standard. VME-based instrumentation modules were connected via RS-232 serial to a PC, which was also on the ethernet network (Spataro and Kent, 1998).

The system implemented has provided fast and reliable data communication between the host computer and the instrumentation modules. Software was developed in C programming language using the User Datagram Protocol (UDP) to enable data packet sending and receiving between the host computer and instrumentation modules. Currently, VXI communication is being explored as a potential replacement for the VME-based instrumentation modules and software has also been developed for this purpose.

Several software packages including DATAIN, COMFRC (Edwards *et al*, to be published), MONFRC (Edwards, 1999), CALIB (Leung and Link, 1999), and ACTUATOR (Toffoletto, 1999), have also been developed in C and X/Motif languages. These packages assist users in the configuration of wind tunnel tests, monitoring of parameters, and the acquisition and reduction of test data. The algorithms of DATAIN, COMFRC and MONFRC were all based on the original Fortran software developed on the old host computer (Fairlie, 1985).

The source code written for the applications above is located on the LSWT host computer denoted as "Bernoulli", and descriptions of all functions are documented in this report. The function summaries are also located on the Defence Science and Technology Organisation (DSTO) Intranet at the following URL location:

<u>http://bernoulli.dsto.defence.gov.au/data_acq/Software/Files/Index.html</u>.

2. Software Description

The concept of the data acquisition system in the LSWT is illustrated in Figure 1. The Digital AlphaServer host computer is connected via a 16-port ethernet hub to PC-based instrumentation modules, which are connected directly to the ethernet using standard network cards. A multi-port serial hub PC acts as the gateway to the VME-based modules by processing ethernet requests and converting then to serial requests.

Each instrumentation module performs a particular function as part of the overall data acquisition system and is connected to its own sensors, motors, controllers, and instrumentation using a range of hardware.

All of the software described in this report resides on the host computer.
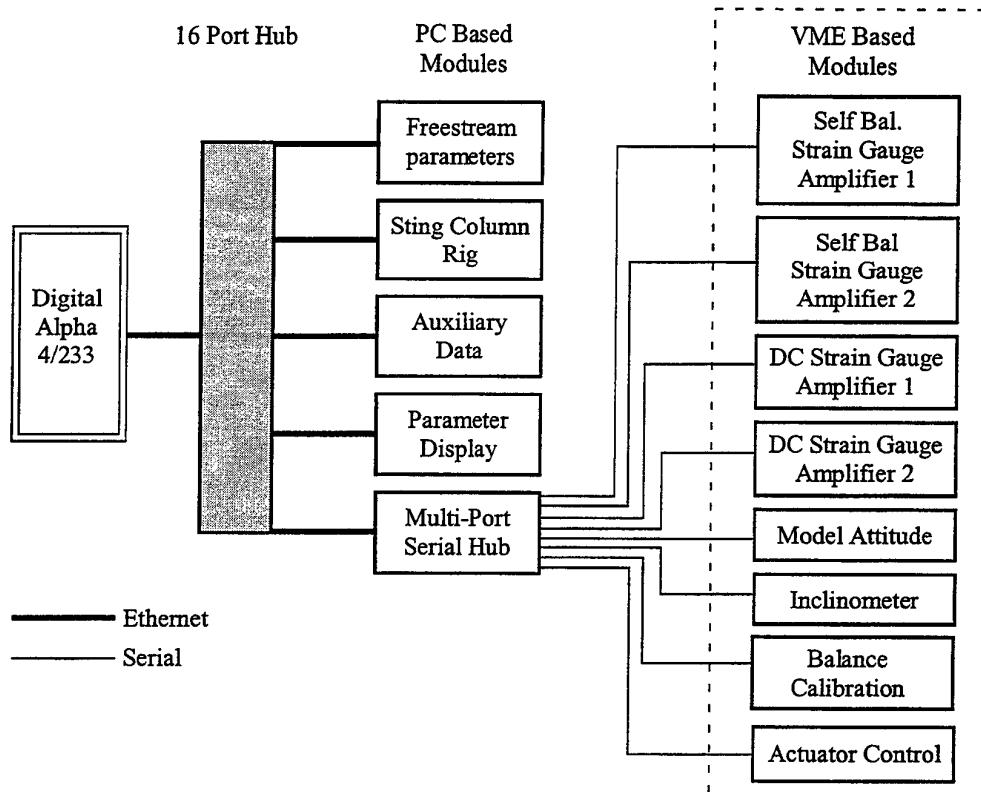
Figure 1. Data acquisition and instrumentation conceptual layout

2.1 DATAIN

DATAIN is the startup graphical interface from which wind tunnel operators can launch applications in a job specific area. On-line or Off-line wind tunnel tests (COMFRC) can be started, actuators can be calibrated and/or controlled (ACTUATOR), configuration data can be entered using WTSETUP (Edwards, 1999) and low-level data communication testing can be performed. DATAIN function summaries are listed in Appendix A.

2.2 COMFRC

COMFRC (Edwards *et al*, to be published) is the primary software for the acquisition and reduction of wind tunnel test data to a form required by the test engineer. It consists of a graphical user interface from which an operator can follow a test schedule and take data accordingly. Aerodynamic coefficients are calculated in real time and are displayed on the screen when data is acquired. Wind tunnel data from previous tests may also be re-processed and displayed *off-line* using this program. It is also capable of reprocessing wind tunnel data obtained from the ASE2000 system in the Transonic Wind Tunnel (TWT). COMFRC uses the UDP communication libraries to acquire the latest data from all of the instrumentation modules. COMFRC function summaries are listed in Appendix B.

2.3 MONFRC

Wind tunnel parameters and data can be displayed graphically in real time using MONFRC (Edwards, 1999). This software is written in C with OpenGL libraries to provide a user-friendly aesthetic interface. Due to the graphics card requirements, MONFRC can only be displayed on the LSWT host computer console screen. MONFRC uses the UDP communication libraries to acquire the latest data from all of the instrumentation modules. MONFRC function summaries are listed in Appendix C.

2.4 CALIB

CALIB is a software package developed in C and X/Motif and it is used to produce the calibration matrix for a force and moment strain gauge balance. The user can choose from three different calibration models and several different order calibration equations. Different methods of least squares fitting, data optimisation and statistical analysis can also be chosen. CALIB function summaries are listed in Appendix D.

2.5 ACTUATOR

Many wind tunnel models include control surfaces that can be remotely moved and controlled using actuators. The software ACTUATOR (Toffoletto, 1999) enables wind tunnel operators to control and calibrate these actuators via a graphical user interface. ACTUATOR uses the UDP communication libraries to acquire data from the actuator

module and also to send commands to the actuator module. ACTUATOR function summaries are listed in Appendix E.

2.6 UDP Communication

This library of functions enables UDP data packets to be constructed and sent from the host computer to the instrumentation modules or vice-versa. A shared memory model on the host computer was used as the interprocess communication to enable multiple software programs to access the most current data. UDP communication library function summaries are listed in Appendix F.

2.7 VXI Communication

This library of functions enables TCP/IP data packets to be received from the VXI controller PC by the LSWT host computer, and places them in a shared memory area providing other software programs with access to the data. VXI communication library function summaries are listed in Appendix G.

3. Outline of the Function Summaries

The function summaries for all software packages described in Section 2 are listed in the Appendices. They are intended as a reference and as a guide for future software developers of the LSWT data acquisition system. Each function summary has the following format:

FUNCTION	function_name	
PURPOSE	A brief description of the purpose of the function	
SYNTAX	`type function_name(type argument)`	
	`List of the function arguments and their types`	Brief descriptions of the function arguments
	...	...
INCLUDES	`List of "#include" files required for the function's compilation and operation.`	
RETURN VALUES	A list of the possible return values of the function and their significance.	
DESCRIPTION	A detailed description of what the function does, the context, what other functions are called within the function, and the significance of any return values.	

Please note that the following conditions apply to the format of the function summaries in the Appendices:

1. The functions have been listed in alphabetical order within each file. The files are listed in alphabetical order for each software program.
2. Not all lines in the function summary are required and in these cases they have not been included.
3. Simple functions do not have a detailed description section if they are adequately summarised with the other information.
4. Common word abbreviations have been used in the interest of brevity.
5. Grammatical abbreviations have been used in the interest of brevity.
6. Symbols, axes systems and wind tunnel corrections that are referred to in the function summaries are defined in corresponding references.

4. Conclusion

Several software packages have been developed during the implementation of a new data acquisition system in the Low Speed Wind Tunnel at the Aeronautical and Maritime Research Laboratory. Descriptions of the source code functions have been written for reference and are attached as Appendices in this report. The function summaries have also been published on the Defence Science Technology Organisation Intranet and can be found at the following URL location:

http://bernoulli.dsto.defence.gov.au/data_acq/Software/Files/Index.html.

5. References

Edwards, C., Lam, S., Link, Y., (to be published), *COMFRC: Data Acquisition in the AMRL Low Speed Wind Tunnel*, DSTO Aeronautical and Maritime Research Laboratory, Australia.

Edwards, C., (1999), *MONFRC: Software for the Graphical Display of Parameters and Data in the Low Speed Wind Tunnel Data Acquisition System*, DSTO-TN-0218, AR-011-056, DSTO Aeronautical and Maritime Research Laboratory, Australia.

Edwards, C., (1999), *WTSETUP: Software for Creating and Editing Configuration Files in the Low Speed Wind Tunnel Data Acquisition System*, DSTO-TN-0217, AR-011-055, DSTO Aeronautical and Maritime Research Laboratory, Australia.

Fairlie, B.D., (1985), *Algorithms for the Reduction of Wind Tunnel Data Derived from Strain Gauge Force Balances*, Aerodynamics Report 154, AR-004-017, DSTO Aeronautical Research Laboratories, Australia.

Holland, O., Lam, S., Link, Y., (1999), *A New Data Acquisition System for the AMRL Low Speed Wind Tunnel*, DSTO-TR-0896, AR-011-129, DSTO Aeronautical and Maritime Research Laboratory, Australia.

Leung, S.Y.F., Link, Y. Y., (1999), *Comparison and Analysis of Strain Gauge Balance Calibration Matrix Mathematical Models*, DSTO-TR-0857, AR-011-051, DSTO Aeronautical and Maritime Research Laboratory, Australia.

Spataro, M., Kent, S., (1998), *A Serial Communication Interface for Data Acquisition Instrumentation in a Wind Tunnel*, DSTO-TR-0740, AR-010-669, DSTO Aeronautical and Maritime Research Laboratory, Australia.

Toffoletto, R., (1999), *A New Actuator Control Program for the Low Speed Wind Tunnel*, DSTO-TN-0188, AR-010-841, DSTO Aeronautical and Maritime Research Laboratory, Australia.

Appendix A: DATAIN Software Functions

A.1. DATAIN Software Function Index

FUNCTION	DESCRIPTION	FILE
ExitDatain	Exit program datain	datain.c
Help	"Help" call back (not implemented)	datain.c
PopupMessage	Pop up a warning message	datain.c
PutControlButtons	Place the menu pushbuttons onto the main window	datain.c
PutJobDirName	Put the Job Directory name as a label on the title_block widget	datain.c
PutHeadings	Put the LSWT logo and DATAIN program title at the top of the main window	datain.c
RunProgram	Execute a program in the $HOME/bin directory	datain.c
CreateDirDialogCB	A call back function for the CreateDirDialog dialogue box. It creates a new directory.	opendir.c
CreateJobDirectory	Create a new Job Directory with the proper directory structure	opendir.c
OpenDirDialogCB		opendir.c
OpenJobDirectory	Create a new Job Directory with the proper directory structure	opendir.c

A.2. DATAIN Software Function Summaries

FUNCTION	ExitDatain	
PURPOSE	Exits Datain	
SYNTAX	`void ExitDatain(Widget w, XtPointer client_data,` `XmPushButtonCallbackStruct *cbs);`	
	`Widget w;`	ID of the parent widget
INCLUDES	`#include "datain.h"`	

FUNCTION	Help	
PURPOSE	"Help" button callback (currently not implemented)	
SYNTAX	`void Help(Widget w, XtPointer client_data,` `XmPushButtonCallbackStruct *cbs);`	
	`Widget w;`	ID of the parent widget
INCLUDES	`#include "datain.h"`	
DESCRIPTION	Currently not implemented	

FUNCTION	PopupMessage	
PURPOSE	Pop up a warning message	
SYNTAX	`void PopupMessage(Widget w, char *message);`	
	`Widget w;`	ID of the parent widget
	`char *message;`	string containing the message
INCLUDES	`#include "datain.h"`	
DESCRIPTION	This is a simple function to pop up a warning message	

FUNCTION	PutControlButtons	
PURPOSE	Place the menu pushbuttons onto the main window	
SYNTAX	void PutControlButtons(Widget main_w, int numpb, ControlButton *pb, char *header);	
	Widget main_w;	the main window widget
	int numpb;	the number of pushbuttons in the set (should be limited to a maximum of 5)
	ControlButton *pb;	Pointer to an array of pushbuttons declared in the ControlButton structure
	char *header	Pointer to a string containing the title of the present set of pushbuttons.
INCLUDES	#include "datain.h"	

DESCRIPTION	The properties of pushbutton are declared in a structure ControlButton:
	 typedef struct {
	   Widget w; /* Widget ID of the pushbutton */
	   char *name; /* Widget name of the pushbutton */
	   char *label; /* Pushbutton Label */
	   void (*callback)(); /* Pointer to the callback routine */
	 } ControlButton;
	 Each set of pushbuttons are grouped into an array and passed onto the function to be placed in a single row. For this reason, maximum number of pushbuttons (to be placed comfortably in a single row) declared in a set should not be more than 5.	

FUNCTION	PutJobDirName	
PURPOSE	Put the Job Directory name as a label on the title_block widget	
SYNTAX	void PutJobDirName(char *jobdir);	
	char *jobdir;	string containing the name of the job directory
INCLUDES	#include "datain.h"	
DESCRIPTION	This function is called whenever the user opens a job directory. It first checks if the jobdirname widget exists. If it doesn't exist the function creates it, else the text value of the jobdirname widget is set to that of the jobdir.	

FUNCTION	PutHeadings	
PURPOSE	Put the LSWT logo and DATAIN program title at the top of the main window	
SYNTAX	`void PutHeadings(Widget main_w, Display *dpy);`	
	`Widget main_w;`	the main window widget
	`Display *dpy;`	the display context obtained by calling `dpy = XtDisplay(toplevel);` where `toplevel` is the widget returned by `XtVaAppInitialize`
INCLUDES	`#include "datain.h"`	
DESCRIPTION	This function puts the LSWT logo and DATAIN program title at the top of the main window. The LSWT logo is input from the image file "lswtlogo.xpm" located in the $HOME/bin directory. The DATAIN program title is hard coded in the "`#define PROG_TITLE`" statement contained in `datain.h`	

FUNCTION	RunProgram	
PURPOSE	Execute a program in the $HOME/bin directory	
SYNTAX	`void RunProgram(Widget w, char *prog, char *param);`	
	`Widget w;`	ID of the parent widget
	`char *prog;`	name of the program to be executed
	`char *param;`	command line parameters needed by the program, `NULL` if there is none
INCLUDES	`#include "datain.h"`	
DESCRIPTION	The program to be executed should be in the `$HOME/bin` directory. If the program is not found, a warning message will pop up.	

FUNCTION	CreateDirDialogCB	
PURPOSE	A call back function for the CreateDirDialog dialog box. It creates a new directory.	
SYNTAX	`void CreateDirDialog(widget w, int client_data, XmSelectionBoxCallbackStruct *cbs;`	
	`widget w`	ID of the parent widget
	`int client_data`	callback client data
	`XmSelectionBoxCallbackStruct *cbs`	callback structure information
INCLUDES	`#include "datain.h"`	
DESCRIPTION	Creates a job directory and the copies the `template` directory structure and files to the new the job directory if OK is pressed on the dialog box.	

FUNCTION	CreateJobDirectory	
PURPOSE	Create a new Job Directory with the proper directory structure	
SYNTAX	`int CreateJobDirectory(Widget w, int client_data, XmPushButtonCallbackStruct *cbs);`	
	`Widget w;`	ID of the parent widget
	`int client_data;`	Not used
	`XmPushButtonCallbackStruct *cbs;`	Callback data structure
INCLUDES	`#include "datain.h"`	
DESCRIPTION	To simplify the process of creating a proper directory structure for the wind tunnel job, a directory known as `$HOME/task/template` containing the basic directory structure described in OpenJobDirectory is used as a template, so that when the `CreateJobDirectory` function is called, it simply copies the whole template directory structure to the new job directory. Basic configuration and initialisation files may also be included in this template directory so that they too can be copied to the new job directory.	

FUNCTION	OpenDirDialogCB	
PURPOSE	Opens to the selected job directory	
SYNTAX	`void OpenDirDialogCB(Widget w, int client_data, XmSelectionBoxCallbackStruct *cbs);`	
	`Widget w;`	ID of the parent widget
	`int client_data;`	callback client data
	`XmSelectionBoxCallbackStruct *cbs;`	callback structure information
INCLUDES	`#include "datain.h"`	
DESCRIPTION	Changes the working directory to the selected job directory and calls the function `PutJobDirName`	

FUNCTION	OpenJobDirectory	
PURPOSE	Ask the user to select a Job Directory as the default working directory	
SYNTAX	`int OpenJobDirectory(Widget w, int client_data, XmPushButtonCallbackStruct *cbs);`	
	`Widget w;`	ID of the parent widget
	`int client_data;`	Not used
	`XmPushButtonCallbackStruct *cbs;`	Callback data structure
INCLUDES	`#include "datain.h"`	
DESCRIPTION	Most LSWT Data Acquisition Programs requires a proper directory structure before they can operate correctly. The **OpenJobDirectory** function is therefore necessary to be executed at the start of the **datain** program to set the default working directory to an appropriate one. All wind tunnel job sub-directories are expected to be under the **$HOME/task** directory. Each job directory should contain at least the following sub-directories: • `conf/` • `data/` • `results/` • `misc/` • `sched/` This function is normally executed as a pushbutton callback and hence is not expected to return any value. However, the function may be called in the normal fashion, and returns 0 for success opening of the job directory, else -1 is returned.	

Appendix B: COMFRC Software Functions

B.1. COMFRC Software Function Index

FUNCTION	DESCRIPTION	FILE
calall	Recalculate all data lines using average values of initial and final zeros and print the results	calall.c
calalltwt	To calculate tareweight components	calall.c
frclsf	Calculate the least squares slope A in Y=A*X and the standard error of estimate of A, STERR	calall.c
momlsf	Calculate the least squares estimates of A and B in X1=A*X2 + B*X3 and the standard error of estimate, STERR	calall.c
attmod	Converts raw theta, psi, phi attitudes to alpha and beta and to take account of balance offsets and deflections	calcsfrc.c
balred	Produces balance load components from a 6 component SG balance voltage readings using a second order calibration matrix	calcsfrc.c
invbal	Finds the set of voltage readings R equivalent to a set of component loads F.	calcsfrc.c
procalcs	Calculates thrust and torque coefficients for propeller measurement tests	calcsfrc.c
tarewt	Calculates the tare weight contribution	calcsfrc.c
toaxes	Converts wind axes coefficients to body, stability and missile axes coefficients	calcsfrc.c
towind	Converts balance buoyant loads to body axes coefficients and then to wind axes coefficients	calcsfrc.c

wtcorr	Applies wind tunnel corrections to the data	calcsfrc.c
CreateArrowButtons	Create 2 control buttons for Prev & Next schedule commands	comfrc.c
CreateDataField	Create an area for Data Line outputs	comfrc.c
CreateFilenameBox	Create Data File, Config File and Schedule File Name boxes	comfrc.c
CreateManualButtons	Create 2 control buttons for manual setpoint and move commands	comfrc.c
CreatePushButtons	Create an area for Data Line outputs	comfrc.c
CreateScheduleField	Create an area for Schedule file outputs	comfrc.c
exit_comfrc	Exit function of the comfrc program	comfrc.c
ForceUpdate	Function that forces an update of the screen and flushes all X events and requests	comfrc.c
fully_auto_cb	Callback function if fully auto mode is selected from the option menu	comfrc.c
main	Main function for Comfrc application	comfrc.c
man_setpt	Callback function if the manual setpoint push button is pressed	comfrc.c
manual_cb	Callback function if manual mode is selected from the option menu	comfrc.c
PopupGeneralWarningDialog	Create a pop-up general warning dialog	comfrc.c
PopupGeneralWarningResponse	Callback routine for the response of PopupGeneralWarningDialog()	comfrc.c
ProcessArgv	Processes arguments passed through input line	comfrc.c
ProcessKeyEvent	Function to process Keyboard commands	comfrc.c
semi_auto_cb	Callback function if semi auto mode is selected from the option menu	comfrc.c

SetPushButtonsSensitive	Sets the initial sensitivity of the push buttons according to whether a first or final zero has been taken	comfrc.c
tol_cb	Callback function if "apply tolerances" checkbox is toggled	comfrc.c
CreateMenuBar	Create the Menu Bar, incorporating all the menu commands	comfrcmenu.c
file_cb	Callback function for the "File" menu command	comfrcmenu.c
help_cb	Callback function for the "Help" menu command	comfrcmenu.c
options_cb	Callback function for the "Options" menu command	comfrcmenu.c
tools_cb	Callback function for the "Tools" menu command	comfrcmenu.c
AcquireActuatorData	Reads data array from the Actuator Module	DataAcqFuncs.c
AcquireACStrainGauge1Data	Reads data array from the AC Strain Gauge 1 Module	DataAcqFuncs.c
AcquireAuxiliaryData	Reads data array from the Auxiliary Module	DataAcqFuncs.c
AcquireCalibrationModuleData	Reads data array from the Calibration Module	DataAcqFuncs.c
AcquireFreestreamData	Reads data array from the Freestream Module	DataAcqFuncs.c
AcquireInclinometerData	Reads data array from the Inclinometer Module	DataAcqFuncs.c
AcquireStingColumnRigData	Reads data array from the Sting Column Rig Module	DataAcqFuncs.c
AcquireTurntableData	Reads data array from the Turntable Module	DataAcqFuncs.c
AcquireVXIStrainGaugeData	Reads data array from the Strain Gauge Module through the VXI system	DataAcqFuncs.c
ConvertActuatorData	Converts a raw data array obtained from the Actuator module into engineering units and places in the current line pointer	DataAcqFuncs.c

15

ConvertACStrainGauge1Data	Converts a raw data array obtained from the AC Strain Gauge 1 module into engineering units and places in the current line pointer	DataAcqFuncs.c
ConvertAuxiliaryData	Converts a raw data array obtained from the Auxiliary module into engineering units and places in the current line pointer	DataAcqFuncs.c
ConvertCalibrationModuleData	Converts a raw data array obtained from the Calibration module into engineering units and places in a data array of type double	DataAcqFuncs.c
ConvertFreestreamData	Converts a raw data array obtained from the Freestream module into engineering units and places in the current line pointer	DataAcqFuncs.c
ConvertInclinometerData	Converts a raw data array obtained from the Inclinometer module into engineering units and places in the current line pointer	DataAcqFuncs.c
ConvertStingColumnRigData	Converts a raw data array obtained from the Sting Column Rig module into engineering units and places in the current line pointer	DataAcqFuncs.c
ConvertTurntableData	Converts a raw data array obtained from the Turntable module into engineering units and places in the current line pointer	DataAcqFuncs.c
validate_data_not_stale	Ensure that the data obtained is non-stale (i.e. a new packet)	DataAcqFuncs.c
validate_vxi_data_not_stale	Ensure that the data obtained is non-stale (i.e. a new packet)	DataAcqFuncs.c
dsp_data	Send a line of data to the serial line	display.c
dsp_endblock	Send the "end of block" signal	display.c

dsp_init	Initialises the serial line and send the initial strings	display.c
dsp_open	Open the terminal line for I/O	display.c
dsp_reject	Send the "reject" signal	display.c
dsp_send	Send a character to the terminal device	display.c
get_act_channel	Obtain the actuator channels that controls the elevator and rudder	display.c
get_dbvalue	Returns the value of the variable indicated by dbname	display.c
CancelCallback	Call back routine for the Cancel push button on the SetUpActuatorsDialog	filedialog.c
CheckBoxCallback	Callback routine for the "Use different Config File" CheckBox widget	filedialog.c
CreateButton	Create pushbutton controls for the File Dialog	filedialog.c
CreateCheckBox	Establish a checkbox widget for "Using different Config File"	filedialog.c
CreateFileComboBox	Create File Combo Box for the File Dialog	filedialog.c
DataFileCallback	Callback function for DataFile combo box selection	filedialog.c
GetFileList	Get a list of all files from a given sub-directory	filedialog.c
OpenFileCancel	Callback function for the "CANCEL" button in Open File Dialog	filedialog.c
OpenFileDialog	Establish an Open File Dialog for comfrc	filedialog.c
OpenFileHelp	Callback function for the "HELP " pushbutton in Open File Dialog	filedialog.c
OpenFileOK	Call back routine for the "OK" pushbutton in Open File Dialog	filedialog.c
ResetCallback	Call back routine for the Reset push button on the SetUpActuatorsDialog	filedialog.c
SetConfigFileBox	Set the contents of the Config File Combo Box	filedialog.c

SetUpActuatorsDialog	Creates a dialog to inform the user of the setup progress of the actuators	filedialog.c
trim_trailing_spaces	Trim off Trailing spaces from a given string	filedialog.c
Find_Fmt	Searches for String in file fp, then returns file pointer to original position	initfrc.c
FndNextLine	Reads file fp until the next valid line is reached.	initfrc.c
Find_Var	Finds a specified string within a specified file	initfrc.c
init	Does initialisation for wind tunnel test	initfrc.c
ReadCharArray	Splits a string (Line) into an array of sub-strings, which are separated by one or more spaces, or tabs.	initfrc.c
Read_Config_File	Reads all configuration data from the supplied configuration file	initfrc.c
Read_Old_Config_File	Reads all configuration data from the supplied configuration file (of the old format)	initfrc.c
ReadSchedFile	Reads the test schedule information from the specified Test Schedule file into a SchedFilePtr	initfrc.c
Read_Vector	Reads numbers from a string into a vector	initfrc.c
Write_Config_File	Writes configuration data in the ConfigDataPtr to a file	initfrc.c
hostZero	Writes the inclinometer offsets contained in the modules initialisation file down to the Inclinometer Module	initModules.c
initFs	Does initialisation of the Freestream Module	initModules.c
initInc	Does initialisation of the Inclinometer Module	initModules.c
initSCR	Does initialisation of Sting Column Rig	initModules.c

initTurntable	Does initialisation of Turntable Module	initModules.c
selfZero	Performs a "zero" of the inclinometer	initModules.c
setReNoLength	Writes the Reynolds Number length down to the Freestream Module for correct calculation and display of Reynolds number	initModules.c
streamAllModules	Sends a streaming command to all communicable modules	initModules.c
input_offline	Inputs data from a raw data file	inputfrc.c
input_online	Acquires a line of data from the instrumentation modules	inputfrc.c
CheckStatus	This is the timeout procedure for the moving dialog, which continually updates the current wind tunnel parameter values and the movement status indicators. It also indicates whether current values are within tolerance of the test schedule file setpoints	mdl_cntrl.c
close_cb	This is the Close button callback for the moving dialog	mdl_cntrl.c
move_cb	This is the Move button callback for the moving dialog	mdl_cntrl.c
move_online	Creates a dialog on the Comfrc interface to show test schedule file parameter setpoints, current values and movement status	mdl_cntrl.c
stop_cb	This is the Stop button callback for the moving dialog	mdl_cntrl.c
wprint	Writes formatted output to text widgets	misc.c
average	Finds the average of the first and final zeros and calculates the buoyant zeros	normalfrc.c
chekit	Checks zero lines for consistency of attitude	normalfrc.c
convert_data	Converts one raw data line to engineering units	normalfrc.c

FindFileConstIndex	Finds whether the specified file constant exists in the `ConfigDataPtr`	normalfrc.c
hextodec	Converts 2 word parameters to the correct integer format	normalfrc.c
normal	Processes a normal data line	normalfrc.c
printout	Outputs to the screen the current line vales computed from the raw data values	normalfrc.c
sgdata	Converts raw strain gauge data to a voltage	normalfrc.c
sortit	Determines whether a line is a zero line or a normal force data line	normalfrc.c
CalculateX1invX2	Calculates the X1inv matrix and performs the matrix multiplication X1inv * X2	offlineASE.c
ConvertDeflConstFormat	Converts the deflection matrix read in from the ASE2000 TWT data file to the LSWT format	offlineASE.c
ConvertDMatrixAxesFormat	Converts the D matrix read in from the ASE2000 TWT data file (X,Y,Z,l,m,n format) to the LSWT format (X,l,Y,m,Z,n)	offlineASE.c
ConvertTaresFormat	Converts the tares matrix read in from the ASE2000 TWT data file to the LSWT format	offlineASE.c
ConvertX1MatrixAxesFormat	Converts the X1 matrix read in from the ASE2000 TWT data file (X,Y,Z,l,m,n format) to the LSWT format (X,l,Y,m,Z,n)	offlineASE.c
ConvertX2MatrixAxesFormat	Converts the X2matrix read in from the ASE2000 TWT data file (X,Y,Z,l,m,n format) to the LSWT format (X,l,Y,m,Z,n)	offlineASE.c
offlineASE	Processes ASE2000 transonic wind tunnel data offline	offlineASE.c
offlineASEfrc	Reads and processes ASE2000 TWT force and moment data files offline	offlineASE.c
offlineASEtwt	Reads and processes ASE2000 TWT tareweight data offline	offlineASE.c

Read_Config_ASE	Reads in the configuration data from both the ASE2000 transonic wind tunnel data file and a LSWT configuration file	offlineASE.c
ReadParameterMatrix	Searches for the header in the ASE2000 TWT data file and then reads in the matrix on the following lines according to the number of rows and columns specified	offlineASE.c
ReadParameterString	Searches for the header in the ASE2000 TWT data file and then reads in the string on the following line	offlineASE.c
ReadParameterValue	Searches for the header in the ASE2000 TWT data file and then reads in the value on the following line	offlineASE.c
comfrc_offline	Performs off-line computation of raw data for Comfrc	offlinefrc.c
read_dataline_offline	Reads a line of raw data knowing that it is only data. Called in the offline sampling case	offlinefrc.c
sample_data_offline	Performs the processing of the data in the off-line case if sampling is required. It includes statistical computations	offlinefrc.c
add_to_list	Allocates memory for another pointer after the last in the linked list and moves to that pointer	onlinefrc.c
calculate_stats	Calculates the statistical information for the samples at each data point	onlinefrc.c
comfrc_online	Performs on-line computation of raw data for Comfrc	onlinefrc.c
count	Returns the number of structures in the list pointed to by head	onlinefrc.c
cumulate_stats	Cumulates the statistical information for the samples at each data point	onlinefrc.c

finalz_online	Acquires a final zero data line, sets the corresponding flag and reprocesses all data based on the first and final zero data	onlinefrc.c
find_penultimate	Returns the offset (file position) of the penultimate line of a file	onlinefrc.c
firstz_online	Acquires a first zero data line and sets the corresponding flag	onlinefrc.c
get_data_online	Acquires data from the instrumentation modules, processes it and writes the raw data to a file	onlinefrc.c
help_online	Brings up available help documentation (currently not implemented)	onlinefrc.c
history_online	Performs history plot on dspfrc (currently not implemented)	onlinefrc.c
init_block	Initialises a new block in Comfrc - resetting of flags	onlinefrc.c
next_online	Prepares for the next setpoint in the test schedule file	onlinefrc.c
normaldata_to_widget	Outputs results from normal to a text widget on the Comfrc application interface	onlinefrc.c
prev_online	Prepares for the previous setpoint in the test schedule file	onlinefrc.c
reject_online	Rejects the last line of data that was acquired	onlinefrc.c
sample_data_online	Performs the sampling and processing of the data in the on-line case. It includes statistical computations	onlinefrc.c
stdev	Calculates the standard deviation from the sum, the sum of the squares and the number of samples	onlinefrc.c
string_to_widget	Outputs a string to a text widget on the Comfrc application interface	onlinefrc.c
TakeDataSamples	Obtains data samples and displays a working dialog on the Comfrc interface to indicate the progress of data acquisition	onlinefrc.c

write_rawdata_line	Writes a single line of acquired raw data to the raw data file	onlinefrc.c
prtini	Prints the header information to the results output file and statistics file (if sampling required) for a force and moment test	prtout.c
prtini_twt	Prints the header information to the results output file and statistics file (if sampling required) for a tareweight calculation	prtout.c
prtblk	Outputs a blank line to the output results file and statistics file (if sampling required) to indicate a new block	prtout.c
prtline	Outputs a line of processed data to the output results file according to the output format defined in the `ConfigDataPtr`	prtout.c
prtstdev	Outputs a line of standard deviations of certain parameters for each group of samples	prtout.c
AddPaddingSpaces	Adds padding spaces to space out the header items above the test schedule field	schedule.c
InputScheduleList	Opens the test schedule file and loads in the header items and parameter values to the test schedule field on the Comfrc interface	schedule.c
SetSchedulePosition	Increments the test schedule file list by the increment amount and highlights the current position	schedule.c

B.2. COMFRC Software Function Summaries

FUNCTION	calall	
PURPOSE	Recalculate all data lines using average values of initial and final zeros and print the results	
SYNTAX	`int calall(GlobalPtr gp)`	
	`GlobalPtr gp`	Global Comfrc pointer

INCLUDES	`#include "calallfrc.h"`
	`#include "debugs.h"`
	`#include "prtout.h"`	

DESCRIPTION	This function traverses through the linked list starting at the first data line after the first_zero line, and using the 'first' variable as the traversing pointer, all valid data lines are re-calculated. 1. Balance voltages are corrected for buoyant zeros 2. Balance loads are calculated using calibration matrices (calls `balred` function) 3. Aerodynamic loads calculated by removing tareweight component (calls `tarewt` function) 4. Calculate alpha/beta attitudes and correct for deflections (calls `attmod` function) 5. Converts balance buoyant loads to body then wind axes (calls `towind` function) 6. Applies wind tunnel corrections (calls `wtcorr` function) 7. Converts wind axes coefficients to body, stability and missile axes (calls `toaxes` function) 8. Prints dataline to output file (calls `prtline` function)	

FUNCTION	calalltwt	
PURPOSE	To calculate tareweight components	
SYNTAX	int calalltwt(GlobalPtr gp)	
	GlobalPtr gp;	Global Comfrc pointer
INCLUDES	#include "comfrc_header.h" #include "calalltwt.h" #include "calcsfrc.h" #include "calallfrc.h" #include "debugs.h" #include "prtout.h"	
DESCRIPTION	This function assumes a linear (first-order) balance calibration and hence can calculate the tare weight components (using a least squares fit) directly without iteration. However because model attitudes are generally derived from sting base values and must be corrected for sting and balance deflections, iteration is required. This function traverses the linked list starting at the first data line after the first_zero data line, and using the 'first' variable as the traversing pointer, all valid data lines are re-calculated. 1. Calculate tare loads from present estimates 2. Get balance readings equivalent to these loads (calls invbal function) 3. Calculate buoyant zeros 4. Calculate component loads (calls balred function) 5. Calculate sums with deflection corrected attitudes 6. Calculate least squares fits to forces and moments and check for convergence 7. Iterate or output tareweight values to file	

FUNCTION	frclsf	
PURPOSE	Calculate the least squares slope A in Y=A*X and the standard error of estimate of A, STERR	
SYNTAX	`int frclsf(double nn, double x, double y, double xx, double yy, double xy, double *a, double *sterr)`	
	`double nn`	number of points
	`double x`	sum of x
	`double y`	sum of y
	`double xx`	sum of x*x
	`double yy`	sum of y*y
	`double xy`	sum of x*y
	`double *a`	slope A
	`double *sterr`	standard error

INCLUDES	`#include "comfrc_header.h"`
	`#include "calalltwt.h"`
	`#include "calcsfrc.h"`
	`#include "calallfrc.h"`
	`#include "debugs.h"`
	`#include "prtout.h"`	

DESCRIPTION	Calculates the least squares slope and standard error in Y= A*X using the various sums of the x and y variables	

FUNCTION	momlsf
PURPOSE	Calculate the least squares estimates of A and B in X1=A*X2 + B*X3 and the standard error of estimate, STERR
SYNTAX	`int momlsf(double nn, double x1, double x2, double x3, double x1x1, double x2x2, double x3x3, double x1x2, double x2x3, double x1x3, double *a, double *b, double *sterr)`

	`double nn`	number of points
	`double x1`	sum of x1
	`double x2`	sum of x2
	`double x3`	sum of x3
	`double x1x1`	sum of x1*x1
	`double x2x2`	sum of x2*x2
	`double x3x3`	sum of x3*x3
	`double x1x2`	sum of x1*x2
	`double x2x3`	sum of x2*x3
	`double x1x3`	sum of x1*x3
	`double *a`	slope A
	`double *b`	slope B
	`double *sterr`	standard error

INCLUDES	`#include "comfrc_header.h"`
	`#include "calalltwt.h"`
	`#include "calcsfrc.h"`
	`#include "calallfrc.h"`
	`#include "debugs.h"`
	`#include "prtout.h"`

DESCRIPTION	Calculates the least squares estimates and standard error in X1= A*X2 + B*X3 using the various sums of the x1, x2 and x3 variables

FUNCTION	attmod	
PURPOSE	Converts raw theta, psi, phi attitudes to alpha and beta and to take account of balance offsets and deflections	
SYNTAX	`int attmod(ConfigDataPtr cdp, CurrentLinePtr clp)`	
	`ConfigDataPtr cdp`	Configuration Data pointer
	`CurrentLinePtr clp`	Current Line pointer
INCLUDES	`#include "comfrc_header.h"` `#include "debugs.h"`	
DESCRIPTION	This function first calculates balance/sting deflections from balance buoyant loads and corrects for theta and phi offsets. If the turntable is the source of psi data, beta is obtained from this. Otherwise alpha and beta are calculated from the deflected attitudes.	

FUNCTION	balred	
PURPOSE	Produces balance load components from a 6 component SG balance voltage readings using a second order calibration matrix	
SYNTAX	`int balred(ConfigDataPtr cdp, double f[6], double r[6])`	
	`ConfigDataPtr cdp`	Configuration Data pointer
	`double f[6]`	vector of component loads
	`double r[6]`	vector of balance voltages
INCLUDES	`#include "comfrc_header.h"` `#include "debugs.h"`	
DESCRIPTION	This function produces balance load components from a 6 component SG balance voltage readings using a second order calibration matrix. First order approximation to F is calculated using calibration matrix data contained in `ConfigDataPtr`. A vector of squares and cross-products of F is formed and the non-linear contribution is calculated. Approximation to F is updated and convergence is checked.	

FUNCTION	invbal	
PURPOSE	Finds the set of voltage readings R equivalent to a set of component loads F.	
SYNTAX	`int invbal(ConfigDataPtr cdp, double fb[6], double rb[6])`	
	`ConfigDataPtr cdp`	Configuration Data pointer
	`double fb[6]`	vector of component loads
	`double rb[6]`	vector of balance voltage outputs at unit excitation
INCLUDES	`#include "comfrc_header.h"` `#include "debugs.h"`	
DESCRIPTION	This function calculates the set of voltage readings equivalent to a set of component loads. A vector of squares and cross-products of loads is formed. The non-normalised reading components are calculated using the calibration matrix data contained in the `ConfigDataPtr`. Readings are finally calculated by predividing by D (the diagonal matrix contained in the `ConfigDataPtr`).	

FUNCTION	procalcs	
PURPOSE	Calculates thrust and torque coefficients for propeller measurement tests	
SYNTAX	`int procalcs(ConfigDataPtr cdp, CurrentLinePtr clp)`	
	`ConfigDataPtr cdp`	Configuration Data pointer
	`CurrentLinePtr clp`	Current Line pointer
INCLUDES	`#include "comfrc_header.h"` `#include "debugs.h"`	
DESCRIPTION	This function calculates the thrust and torque coefficients and advance ratio.	

FUNCTION	tarewt
PURPOSE	Calculates the tare weight contribution
SYNTAX	`int tarewt(ConfigDataPtr cdp, CurrentLinePtr clp)`
	`ConfigDataPtr cdp`	Configuration Data pointer
	`CurrentLinePtr clp`	Current Line pointer
INCLUDES	`#include "comfrc_header.h"` `#include "debugs.h"`
DESCRIPTION	This function calculates the tareweight contribution and uses the same algorithm whether the model is rear-sting mounted or mounted via a pylon and rear spear. Tareweight contribution is added to component loads using model attitudes. Conversion of units is performed depending on the form of the calibration matrix.

FUNCTION	toaxes
PURPOSE	Converts wind axes coefficients to body, stability and missile axes coefficients
SYNTAX	`int toaxes(ConfigDataPtr cdp, CurrentLinePtr clp)`
	`ConfigDataPtr cdp`	Configuration Data pointer
	`CurrentLinePtr clp`	Current Line pointer
INCLUDES	`#include "comfrc_header.h"` `#include "debugs.h"`
DESCRIPTION	This function converts wind axes coefficients to body, stability and missile axes coefficients using the alpha and beta attitudes and psi obtained from `attmod`.

FUNCTION	towind	
PURPOSE	Converts balance buoyant loads to body axes coefficients and then to wind axes coefficients	
SYNTAX	`int towind(ConfigDataPtr cdp, CurrentLinePtr clp)`	
	`ConfigDataPtr cdp`	Configuration Data pointer
	`CurrentLinePtr clp`	Current Line pointer
INCLUDES	`#include "comfrc_header.h"` `#include "debugs.h"`	
DESCRIPTION	This function first converts the balance buoyant loads to balance axes coefficients. The balance axes coefficients are converted to body axes coefficients using the offsets located in the `ConfigDataPtr`. The body axes coefficients are then converted to wind axes coefficients using the alpha and beta attitudes obtained from `attmod`.	

FUNCTION	wtcorr	
PURPOSE	Applies wind tunnel corrections to the data	
SYNTAX	`int wtcorr(ConfigDataPtr cdp, CurrentLinePtr clp)`	
	`ConfigDataPtr cdp`	Configuration Data pointer
	`CurrentLinePtr clp`	Current Line pointer
INCLUDES	`#include "comfrc_header.h"` `#include "debugs.h"`	
DESCRIPTION	This function applies wind tunnel corrections to the data including blockage, lift interference factors, tailplane angle and alpha corrections. Refer to: Fairlie, B.D., (1985), *Algorithms for the Reduction of Wind Tunnel Data Derived from Strain Gauge Force Balances*, Aerodynamics Report 154, AR-004-017, DSTO Aeronautical Research Laboratories, Australia.	

FUNCTION	CreateArrowButtons	
PURPOSE	Create 2 control buttons for Prev & Next schedule commands	
SYNTAX	void CreateArrowButtons(Widget parent, ComfrcControlButton *cb, GlobalPtr gp)	
	Widget parent	parent widget
	GlobalPtr gp	Global Comfrc pointer
	ComfrcControlButton *cb	Structure containing information pertaining to buttons on Comfrc interface
INCLUDES	#include "comfrc_header.h" #include "comfrc.h" #include "onlinefrc.h" #include "mdl_cntrl.h" #include "schedule.h" #include "mDB_Indx.h"	
DESCRIPTION	Creates the next and previous arrow buttons for the test schedule field Creates the option menu widget with manual/semi-auto/fully-auto modes of data acquisition using the test schedule.	

FUNCTION	CreateDataField	
PURPOSE	Create an area for Data Line outputs	

SYNTAX	`void`
	`CreateDataField (Widget parent, XWindowsWidgets`
	`*xwp,`
	`LocalDataPtr ldp, XmFontList fontlist)`	

	`Widget parent;`	ID of the parent widget
	`XWindowsWidgets *xwp`	Pointer to a structure containing information on specific widgets in the Comfrc interface
	`LocalDataPtr ldp`	Pointer to structure containing information local to Comfrc such as the filenames and sequencing of a wind tunnel run
	`XmFontList fontlist`	Structure of font types for text display

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "mdl_cntrl.h"`
	`#include "schedule.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	Creates a text field for the output of acquired data lines including wind tunnel parameters and force/moment coefficients	

FUNCTION	CreateFilenameBox
PURPOSE	Create Data File, Config File and Schedule File Name boxes
SYNTAX	void CreateFilenameBox(Widget parent, XwindowsWidgets *xwp)
	Widget parent;	parent widget
	XwindowsWidget s *xwp	Pointer to a structure containing information on specific widgets in the Comfrc interface
INCLUDES	#include "comfrc_header.h" #include "comfrc.h" #include "onlinefrc.h" #include "mdl_cntrl.h" #include "schedule.h" #include "mDB_Indx.h"
DESCRIPTION	Creates Data File, Configuration File and Test Schedule File name text boxes to be input by the user

FUNCTION	CreateManualButtons
PURPOSE	Create 2 control buttons for Manual Setpoint & Move commands
SYNTAX	void CreateManualButtons(Widget parent, ComfrcControlButton *cb, GlobalPtr gp)
	Widget parent	parent widget
	GlobalPtr gp	Global Comfrc pointer
	ComfrcControlButt on *cb	Structure containing information pertaining to buttons on Comfrc interface
INCLUDES	#include "comfrc_header.h" #include "comfrc.h" #include "onlinefrc.h" #include "mdl_cntrl.h" #include "schedule.h" #include "mDB_Indx.h"
DESCRIPTION	Creates the manual setpoint button to allow the user to enter manual setpoints. Creates a move button to allow the user to move model attitudes/actuators to the setpoint currently highlighted in the test schedule field. Creates a checkbox to allow the user to apply tolerances to the test schedule file parameters for data acquisition

FUNCTION	CreatePushButtons	
PURPOSE	Create comfrc command PushButton Controls	
SYNTAX	`void CreatePushButtons(ComfrcControlButton *cb, Widget parent, int nstart, int nend, int nrows, GlobalPtr gp )`	
	`ComfrcControlButton *cb`	structure containing information pertaining to buttons on Comfrc interface
	`Widget parent`	parent widget ID
	`int nstart`	index of first button in cb to be created
	`int nend`	index of last button in cb to be created
	`int nrows`	number of rows over which to create the push buttons
	`GlobalPtr gp`	Global Comfrc pointer
INCLUDES	`#include "comfrc_header.h"` `#include "comfrc.h"` `#include "onlinefrc.h"` `#include "mdl_cntrl.h"` `#include "schedule.h"` `#include "mDB_Indx.h"`	
DESCRIPTION	Creates the push buttons listed in the structure cb according to the start and end indices passed as arguments to the function. These buttons include the Get Data/First/Final Zero/Block End etc on the Comfrc window.	

FUNCTION	CreateScheduleField
PURPOSE	Create an area for Schedule file outputs
SYNTAX	void CreateScheduleField(Widget parent, GlobalPtr gp, ComfrcControlButton *cb, XmFontList fontlist)
	Widget parent	parent widget
	GlobalPtr gp	Global Comfrc pointer
	ComfrcControlButton *cb	Structure containing information pertaining to buttons on Comfrc interface
	XmFontList fontlist	Structure of font types for text display
INCLUDES	#include "comfrc_header.h" #include "comfrc.h" #include "onlinefrc.h" #include "mdl_cntrl.h" #include "schedule.h" #include "mDB_Indx.h"
DESCRIPTION	Creates a text field for the display of test schedule file information

FUNCTION	exit_comfrc
PURPOSE	Exit function of the Comfrc application
SYNTAX	void exit_comfrc(Widget w, GlobalPtr gp)
	Widget w	parent widget ID
	GlobalPtr gp	Global Comfrc pointer
INCLUDES	#include "comfrc_header.h" #include "comfrc.h" #include "onlinefrc.h" #include "mdl_cntrl.h" #include "schedule.h" #include "mDB_Indx.h"
DESCRIPTION	Exits the Comfrc application. Creates warning dialog if final zeros have not been taken. Kills the Monfrc application if it is running.

FUNCTION	ForceUpdate	
PURPOSE	Function that forces an update of the screen and flushes all X events and requests	
SYNTAX	`void ForceUpdate(Widget w)`	
	`Widget w`	ID of a widget on the window to be updated

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "mdl_cntrl.h"`
	`#include "schedule.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	Forces a graphical update of the window and flushes all X requests	

FUNCTION	fully_auto_cb	
PURPOSE	Callback function if fully auto mode is selected from the option menu	
SYNTAX	`void fully_auto_cb(Widget w, GlobalPtr gp)`	
	`Widget w`	parent widget ID
	`GlobalPtr gp`	Global Comfrc pointer

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "mdl_cntrl.h"`
	`#include "schedule.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	If fully auto mode is selected, the test schedule mode variable is set to 2. Button sensitivities are set. Processes the setpoint currently highlighted in the test schedule field (get data/block online)	

FUNCTION	main	
PURPOSE	Main function for Comfrc application	
SYNTAX	`void main(int argc, char **argv)`	
	`int argc`	number of command line arguments
	`char **argv`	command line arguments - online/offline - force/tareweight/propeller -ASE2000 TWT processing

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "mdl_cntrl.h"`
	`#include "schedule.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	Allocate and initialise memories for common variables Setup X-Windows widgets for Comfrc graphical interface Display Open File Dialog	

FUNCTION	man_setpt	
PURPOSE	Callback function if the manual setpoint push button is pressed	
SYNTAX	`void man_setpt(Widget w, GlobalPtr gp)`	
	`Widget w`	parent widget ID
	`GlobalPtr gp`	Global Comfrc pointer

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "mdl_cntrl.h"`
	`#include "schedule.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	If manual setpoint push button is pressed, the function "move_online" is called with the manual setpoint flag set to 1.	

FUNCTION	manual_cb	
PURPOSE	Callback function if manual mode is selected from the option menu	
SYNTAX	`void manual_cb(Widget w, GlobalPtr gp)`	
	`Widget w`	parent widget ID
	`GlobalPtr gp`	Global Comfrc pointer

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "mdl_cntrl.h"`
	`#include "schedule.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	If manual mode is selected, the test schedule mode variable is set to 0. Button sensitivities are set.	

FUNCTION	PopupGeneralWarningDialog	
PURPOSE	Create a pop-up general warning dialog	

SYNTAX	`int PopupGeneralWarningDialog(Widget parent,`
	`char *message, char *ok_label, char *cancel_label,`
	`char *help_label,`
	`XtAppContext app)`	

	`Widget parent`	parent widget ID
	`char *message`	message for the dialog
	`char *ok_label`	label for the OK button
	`char *cancel_label`	label for the Cancel button
	`char *help_label`	label for the Help button
	`XtAppContext app`	Comfrc application context

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "mdl_cntrl.h"`
	`#include "schedule.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	Creates a general warning dialog with a specified message and custom button labels. Modal dialog such that a response must be given before dialog is destroyed.	

FUNCTION	**PopupGeneralWarningResponse**	
PURPOSE	Callback routine for the response of PopupGeneralWarningDialog()	
SYNTAX	`void PopupGeneralWarningResponse(Widget w, int *answer,` `XmAnyCallbackStruct *cbs)`	
	`Widget w`	parent widget ID
	`int *answer`	return answer:
	`XmAnyCallbackStructure *cbs`	Callback structure containing information regarding which button was pressed

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "mdl_cntrl.h"`
	`#include "schedule.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	Sets a value to an integer variable "answer" 1 - OK button pressed 2 - Cancel button pressed 3 - Help button pressed	

FUNCTION	ProcessArgv	
PURPOSE	Processes arguments passed through input line	
SYNTAX	`void ProcessArgv(char *arg_list, GlobalPtr gp)`	
	`char *arg_list`	character line of arguments
	`GlobalPtr gp`	Global Comfrc pointer

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "mdl_cntrl.h"`
	`#include "schedule.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	Processes the switch arguments for Comfrc depending on what type of test is run. Sets local variables: (default is online force and moment test) -o offline -t tareweight, -p propeller run type -a ASE transonic wind tunnel run data to be processed	

FUNCTION	ProcessKeyEvent	
PURPOSE	Function to process Keyboard commands	
SYNTAX	`void ProcessKeyEvent(Widget w, char *key, GlobalPtr gp, ComfrcControlButton *cb)`	
	`Widget w`	parent widget
	`char *key`	key press character
	`GlobalPtr gp`	Global comfrc pointer
	`ComfrcControlButton *cb`	Structure containing information pertaining to buttons on Comfrc interface

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "mdl_cntrl.h"`
	`#include "schedule.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	Processes keyboard commands and calls the appropriate function	

FUNCTION	semi_auto_cb	
PURPOSE	Callback function if semi auto mode is selected from the option menu	
SYNTAX	`void semi_auto_cb(Widget w, GlobalPtr gp)`	
	`Widget w`	parent widget ID
	`GlobalPtr gp`	Global Comfrc pointer

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "mdl_cntrl.h"`
	`#include "schedule.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	If semi auto mode is selected, the test schedule mode variable is set to 1 Button sensitivities are set.	

FUNCTION	SetPushButtonsSensitive	
PURPOSE	Sets the initial sensitivity of the push buttons according to whether a first or final zero has been taken	
SYNTAX	`void SetPushButtonsSensitive(ComfrcControlButton *cb,` `Boolean got_first_zeros)`	
	`ComfrcControlButton *cb`	Structure containing information pertaining to buttons on Comfrc interface
	`Boolean got_first_zeros`	Boolean value true if first zeros have been taken

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "mdl_cntrl.h"`
	`#include "schedule.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	Sets the sensitivity of the first zero button according to the Boolean value passed. Sets the sensitivity of other push buttons according to whether first zeros have been taken.	

FUNCTION	tol_cb	
PURPOSE	Callback function if "apply tolerances" checkbox is toggled	
SYNTAX	`void tol_cb(Widget w, GlobalPtr gp, XtPointer call_data)`	
	`Widget w`	parent widget ID
	`GlobalPtr gp`	Global Comfrc pointer
	`XtPointer call_data`	Callback structure

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "mdl_cntrl.h"`
	`#include "schedule.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	If apply tolerances checkbox is toggled, the flag is set to the appropriate checkbox state according to the information in the callback structure.	

FUNCTION	CreateMenuBar	
PURPOSE	Create the Menu Bar, incorporating all the menu commands	
SYNTAX	`void CreateMenuBar(Widget main_w, GlobalPtr gptr)`	
	`Widget main_w`	mainwindow widget ID
	`GlobalPtr gptr`	Global Comfrc pointer
INCLUDES	`#include "comfrc_header.h"`	
DESCRIPTION	Creates the menu bar and each of the pull down menus on the Comfrc application interface	

FUNCTION	file_cb	
PURPOSE	Callback function for the "File" menu command	
SYNTAX	void file_cb(Widget w, int item_no, XmAnyCallbackStruct *cbs);	
	Widget w;	parent widget ID
	int item_no	item number in pulldown menu
	XmAnyCallbackStruct *cbs	callback structure
INCLUDES	#include "comfrc_header.h"	
DESCRIPTION	Processes the choice made by the user from the File pulldown menu	

FUNCTION	help_cb	
PURPOSE	Callback function for the "Help" menu command	
SYNTAX	void help_cb(Widget w, int item_no, XmAnyCallbackStruct *cbs);	
	Widget w;	parent widget ID
	int item_no	item number in pulldown menu
	XmAnyCallbackStruct *cbs	callback structure
INCLUDES	#include "comfrc_header.h"	
DESCRIPTION	Processes the choice made by the user from the Help pulldown menu	

FUNCTION	options_cb	
PURPOSE	Callback function for the "Options" menu command	
SYNTAX	void options_cb(Widget w, int item_no, XmAnyCallbackStruct *cbs);	
	Widget w;	parent widget ID
	int item_no	item number in pulldown menu
	XmAnyCallbackStruct *cbs	callback structure
INCLUDES	#include "comfrc_header.h"	
DESCRIPTION	Processes the choice made by the user from the Options pulldown menu	

FUNCTION	tools_cb	
PURPOSE	Callback function for the "Tools" menu command	
SYNTAX	`void tools_cb(Widget w, int item_no, XmAnyCallbackStruct *cbs);`	
	`Widget w;`	parent widget ID
	`int item_no`	item number in pulldown menu
	`XmAnyCallbackStruct *cbs`	callback structure
INCLUDES	`#include "comfrc_header.h"`	
DESCRIPTION	Processes the choice made by the user from the Tools pulldown menu	

FUNCTION	AcquireActuatorData	
PURPOSE	Reads data array from the Actuator Module	
SYNTAX	`int AcquireActuatorData(int *data, ActDataPtr adp, int priority)`	
	`int *data`	array of integer data values
	`ActDataPtr adp`	Actuator Data pointer
	`int priority`	priority of data acquisition

INCLUDES	`#include "DataAcqFuncs.h"`
	`#include "normalfrc.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	This function reads the required actuator angles values from the Actuator module into a data array. The valid actuator channels are determined from the `ActDataPtr`. If priority is set =1, then the data obtained from the module must be a new packet (i.e. non-stale data)	

FUNCTION	**AcquireACStrainGauge1Data**	
PURPOSE	Reads data array from the AC Strain Gauge 1 Module	
SYNTAX	`int AcquireACStrainGauge1Data(int *data, int priority)`	
	`int *data`	array of integer data values
	`int priority`	priority of data acquisition

INCLUDES	`#include "DataAcqFuncs.h"`
	`#include "normalfrc.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	This function reads the six voltage components (X, l, Y, m, Z, n) from the AC Strain Gauge 1 module into a data array. If priority is set =1, then the data obtained from the module must be a new packet (i.e. non-stale data)	

FUNCTION	**AcquireAuxiliaryData**	
PURPOSE	Reads data array from the Auxiliary Module	
SYNTAX	`int AcquireAuxiliaryData(int *data, int priority)`	
	`int *data`	array of integer data values
	`int priority`	priority of data acquisition

INCLUDES	`#include "DataAcqFuncs.h"`
	`#include "normalfrc.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	This function reads the RPS tacho reading from the Auxiliary module into a data array. If priority is set =1, then the data obtained from the module must be a new packet (i.e. non-stale data)	

FUNCTION	AcquireCalibrationModuleData	
PURPOSE	Reads data array from the Calibration Module	
SYNTAX	`int AcquireCalibrationModuleData(int *data, int priority)`	
	`int *data`	array of integer data values
	`int priority`	priority of data acquisition

INCLUDES	`#include "DataAcqFuncs.h"`
	`#include "normalfrc.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	This function reads 12 strain gauge voltages from the Calibration module into a data array. If priority is set =1, then the data obtained from the module must be a new packet (i.e. non-stale data)	

FUNCTION	AcquireFreestreamData	
PURPOSE	Reads data array from the Freestream Module	
SYNTAX	`int AcquireFreestreamData(int *data, int priority)`	
	`int *data`	array of integer data values
	`int priority`	priority of data acquisition

INCLUDES	`#include "DataAcqFuncs.h"`
	`#include "normalfrc.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	This function reads total pressure, airspeed, dynamic pressure, temperature, Mach number and Reynolds number from the Freestream module into a data array. If priority is set =1, then the data obtained from the module must be a new packet (i.e. non-stale data)	

FUNCTION	AcquireInclinometerData	
PURPOSE	Reads data array from the Inclinometer Module	
SYNTAX	`int AcquireInclinometerData(int *data, int priority)`	
	`int *data`	array of integer data values
	`int priority`	priority of data acquisition

INCLUDES	`#include "DataAcqFuncs.h"`
	`#include "normalfrc.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	This function reads the theta and phi values from the Inclinometer module into a data array. If priority is set =1, then the data obtained from the module must be a new packet (i.e. non-stale data)	

FUNCTION	AcquireStingColumnRigData	
PURPOSE	Reads data array from the Sting Column Rig Module	
SYNTAX	`int AcquireStingColumnRigData(int *data, int priority)`	
	`int *data`	array of integer data values
	`int priority`	priority of data acquisition

INCLUDES	`#include "DataAcqFuncs.h"`
	`#include "normalfrc.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	This function reads alpha, beta, theta, phi, and height offset from the Sting Column Rig module into a data array. If priority is set =1, then the data obtained from the module must be a new packet (i.e. non-stale data)	

FUNCTION	AcquireTurntableData	
PURPOSE	Reads data array from the Turntable Module	
SYNTAX	`int AcquireTurntableData(int *data, TurntableDataPtr tdp, int priority)`	
	`int *data`	array of integer data values
	`TurntableDataPtr tdp`	Turntable Data pointer
	`int priority`	priority of data acquisition

INCLUDES	`#include "DataAcqFuncs.h"`
	`#include "normalfrc.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	This function reads the required turntable angle values from the Turntable module into a data array. The valid turntables are determined from the `TurntableDataPtr`. If priority is set =1, then the data obtained from the module must be a new packet (i.e. non-stale data)	

FUNCTION	AcquireVXIStrainGaugeData	
PURPOSE	Reads data array from the Strain Gauge Module through the VXI system	
SYNTAX	`int AcquireVXIStrainGaugeData(int *data, int priority)`	
	`int *data`	array of integer data values
	`int priority`	priority of data acquisition

INCLUDES	`#include "DataAcqFuncs.h"`
	`#include "normalfrc.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	This function reads 16 strain gauge voltages from the Strain gauge module through the VXI system into a data array. The VXI system is in the developmental stages and is external to the data acquistion system currently in operation. If priority is set =1, then the data obtained from the module must be a new packet (i.e. non-stale data)	

FUNCTION	ConvertActuatorData	
PURPOSE	Converts a raw data array obtained from the Actuator module into engineering units and places in the current line pointer	
SYNTAX	`void ConvertActuatorData(int *data, CurrentLinePtr clp, ActDataPtr adp)`	
	`int *data`	data array
	`CurrentLinePtr clp`	Current Line pointer
	`ActDataPtr adp`	Actuator Data pointer
INCLUDES	`#include "comfrc_header.h"` `#include "debugs.h"`	
DESCRIPTION	This function converts the data array obtained from the Actuator module (actuator angles according to the valid channels contained in the `ActDataPtr`) into engineering units and places the data into the `CurrentLinePtr`.	

FUNCTION	ConvertACStrainGauge1Data
PURPOSE	Converts a raw data array obtained from the AC Strain Gauge 1 module into engineering units and places in the current line pointer
SYNTAX	`void ConvertACStrainGauge1Data(int *data, CurrentLinePtr clp)`
	`int *data` data array
	`CurrentLinePtr clp` Current Line pointer
INCLUDES	`#include "comfrc_header.h"` `#include "debugs.h"`
DESCRIPTION	This function converts the data array obtained from the AC Strain Gauge 1 module (six strain gauge voltage readings) into engineering units and places the data into the `CurrentLinePtr`.

FUNCTION	ConvertAuxiliaryData
PURPOSE	Converts a raw data array obtained from the Auxiliary module into engineering units and places in the current line pointer
SYNTAX	`void ConvertAuxiliaryData(int *data, CurrentLinePtr clp)`
	`int *data` data array
	`CurrentLinePtr clp` Current Line pointer
INCLUDES	`#include "comfrc_header.h"` `#include "debugs.h"`
DESCRIPTION	This function converts the data array obtained from the Auxiliary module (RPS tacho reading) into engineering units and places the data into the `CurrentLinePtr`.

FUNCTION	ConvertCalibrationModuleData	
PURPOSE	Converts a raw data array obtained from the Calibration module into engineering units and places in a data array of type double	
SYNTAX	`void ConvertCalibrationModuleData(int *data, double *cal_data)`	
	`int *data`	data array
	`double *cal_data`	data array
INCLUDES	`#include "comfrc_header.h"` `#include "debugs.h"`	
DESCRIPTION	This function converts the data array obtained from the Balance Calibration module (12 strain gauge voltage readings) into engineering units and places the data into the array `cal_data`.	

FUNCTION	ConvertFreestreamData	
PURPOSE	Converts a raw data array obtained from the Freestream module into engineering units and places in the current line pointer	
SYNTAX	`void ConvertFreestreamData(int *data, CurrentLinePtr clp)`	
	`int *data`	data array
	`CurrentLinePtr clp`	Current Line pointer
INCLUDES	`#include "comfrc_header.h"` `#include "debugs.h"`	
DESCRIPTION	This function converts the data array obtained from the Freestream module (total pressure, dynamic pressure, airspeed, temperature, Mach number, Reynolds number) into engineering units and places the data into the `CurrentLinePtr`. Air density is also calculated in this function using the total pressure and temperature.	

FUNCTION	ConvertInclinometerData	
PURPOSE	Converts a raw data array obtained from the Inclinometer module into engineering units and places in the current line pointer	
SYNTAX	`void ConvertInclinometerData(int *data, CurrentLinePtr clp)`	
	`int *data`	data array
	`CurrentLinePtr clp`	Current Line pointer
INCLUDES	`#include "comfrc_header.h"` `#include "debugs.h"`	
DESCRIPTION	This function converts the data array obtained from the Inclinometer module (theta and phi angles) into engineering units and places the data into the `CurrentLinePtr`.	

FUNCTION	ConvertStingColumnRigData	
PURPOSE	Converts a raw data array obtained from the Sting Column Rig module into engineering units and places in the current line pointer	
SYNTAX	`void ConvertStingColumnRigData(int *data, CurrentLinePtr clp)`	
	`int *data`	data array
	`CurrentLinePtr clp`	Current Line pointer
INCLUDES	`#include "comfrc_header.h"` `#include "debugs.h"`	
DESCRIPTION	This function converts the data array obtained from the Sting Column Rig module (alpha, beta, theta, phi, height offset) into engineering units and places the data into the `CurrentLinePtr`.	

FUNCTION	ConvertTurntableData	
PURPOSE	Converts a raw data array obtained from the Turntable module into engineering units and places in the current line pointer	
SYNTAX	`void ConvertTurntableData(int *data, CurrentLinePtr clp, TurntableDataPtr tdp)`	
	`int *data`	data array
	`CurrentLinePtr clp`	Current Line pointer
	`TurntableDataPtr tdp`	Turntable Data pointer
INCLUDES	`#include "comfrc_header.h"` `#include "debugs.h"`	
DESCRIPTION	This function converts the data array obtained from the Turntable module into engineering units and places the data into the `CurrentLinePtr`. Turntable offsets are obtained from the `TurntableDataPtr`.	

FUNCTION	validate_data_not_stale	
PURPOSE	Ensure that the data obtained is non-stale (i.e. a new packet)	
SYNTAX	`int validate_data_not_stale(int module, unsigned int *prevCountPtr)`	
	`int module`	Number of the module
	`unsigned int *prevCountPtr`	previous packet capture count
INCLUDES	`#include "comfrc_header.h"` `#include "debugs.h"`	
DESCRIPTION	Checks the current packet capture number and only returns TRUE if the current packet capture number has incremented therefore signalling new data has arrived.	

FUNCTION	validate_vxi_data_not_stale
PURPOSE	Ensure that the data obtained is non-stale (i.e. a new packet)
SYNTAX	`int validate_vxi_data_not_stale(unsigned int *prevCountPtr)`
	`unsigned int *prevCountPtr`	previous packet capture count
INCLUDES	`#include "comfrc_header.h"` `#include "debugs.h"`
DESCRIPTION	Checks the current packet capture number and only returns TRUE if the current packet capture number has incremented therefore signally the data is 'non-stale'.

FUNCTION	dsp_data
PURPOSE	Send a line of data to the serial line
SYNTAX	`int dsp_data(ConfigDataPtr cdp, CurrentLinePtr clp)`
	`ConfigDataPtr cdp;`	Configuration Data pointer
	`CurrentLinePtr clp`	Current Line pointer
INCLUDES	`#include "comfrc_header.h"` `#include "display.h"`
DESCRIPTION	Sends a line of data containing the data block variable/constant and the aerodynamic coefficients.

FUNCTION	dsp_endblock
PURPOSE	Send the "end of block" signal
SYNTAX	`int dsp_endblock(void)`
INCLUDES	`#include "comfrc_header.h"` `#include "display.h"`
DESCRIPTION	Sends the "end of block" signal

FUNCTION	dsp_init	
PURPOSE	Initialises the serial line and send the initial strings	
SYNTAX	`int dsp_init(GlobalPtr gp)`	
	`GlobalPtr gp;`	Global Comfrc pointer
INCLUDES	`#include "comfrc_header.h"` `#include "display.h"`	
DESCRIPTION	Opens the serial line connection to the PC and sends strings to initialise the data block variable and constants.	

FUNCTION	dsp_open	
PURPOSE	Open the terminal line for I/O	
SYNTAX	`int dsp_open(char *line, enum parity parity, int baud)`	
	`char *line;`	name of the line for connection
	`enum parity parity;`	parity
	`int baud;`	baud rate
INCLUDES	`#include "comfrc_header.h"` `#include "display.h"`	
DESCRIPTION	Opens a connection to a PC for I/O	

FUNCTION	dsp_reject
PURPOSE	Send the "reject" signal
SYNTAX	`int dsp_reject(void)`
INCLUDES	`#include "comfrc_header.h"` `#include "display.h"`
DESCRIPTION	Sends a signal to reject the last point sent through the serial connection.

FUNCTION	dsp_send	
PURPOSE	Send a character to the terminal device	
SYNTAX	`int dsp_send(int dsp, char *s)`	
	`int dsp;`	handle to the serial port connection
	`char *s;`	character to send
INCLUDES	`#include "comfrc_header.h"` `#include "display.h"`	
DESCRIPTION	Sends a character to the terminal device	

FUNCTION	get_act_channel	
PURPOSE	Obtain the actuator channels that controls the elevator and rudder	
SYNTAX	`int get_act_channel(ActDataPtr adp, char *db_name)`	
	`ActDataPtr adp;`	Actuator Data pointer
	`char *db_name;`	data block variable name
INCLUDES	`#include "comfrc_header.h"` `#include "display.h"`	
DESCRIPTION	Obtains the actuator channels corresponding to the elevator and rudder by comparing the db_name with DELE and DELR and the actuator description in adp with "Elev" and "Rud"	

FUNCTION	get_dbvalue	
PURPOSE	Returns the value of the variable indicated by dbname	
SYNTAX	`double get_dbvalue(char *dbname, CurrentLinePtr clp, int idx)`	
	`char *dbname;`	data block variable name
	`CurrentLinePtr clp;`	Current Line pointer
	`int idx;`	actuator channel index number
INCLUDES	`#include "comfrc_header.h"` `#include "display.h"`	
DESCRIPTION	Returns the value of the variable in clp indicated by dbname	

FUNCTION	CancelCallback	
PURPOSE	Callback function for Cancel on SetUpActuatorsDialog	
SYNTAX	`void CancelCallback(Widget w, int *finished, XmPushButtonCallbackStruct *cbs)`	
	`Widget w`	parent widget ID
	`int *finished`	return finished value of 1
	`XmPushButtonCallbackStruct *cbs`	callback structure information for cancel push button

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "offlineASE.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	Returns a value of 1 for finished variable to close the dialog	

FUNCTION	CheckBoxCallback	
PURPOSE	Callback routine for the "Use different Config File" CheckBox widget	
SYNTAX	`void CheckBoxCallback (Widget parent, GlobalPtr gp, XmToggleButtonCallbackStruct *toggle_status)`	
	`Widget parent`	widget parent ID
	`GlobalPtr gp`	Global Comfrc pointer
	`XmToggleButtonCallbackStruct *toggle_status`	Callback structure information for the checkbox

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "offlineASE.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	Sets the sensitivity and contents of the config file combo box	

FUNCTION	CreateButton
PURPOSE	Create pushbutton controls for the File Dialog
SYNTAX	```void CreateButton(Widget parent, ControlButtonItem *buttons, int num_buttons)```
	`Widget parent`	parent widget ID
	`ControlButtonItem *buttons`	structure containing information about the buttons of the file dialog
	`int num_buttons`	number of buttons
INCLUDES	```#include "comfrc_header.h" #include "comfrc.h" #include "onlinefrc.h" #include "offlineASE.h" #include "mDB_Indx.h"```
DESCRIPTION	Creates the push buttons of the file dialog and sets up callbacks

FUNCTION	CreateCheckBox
PURPOSE	Establish a checkbox widget for "Using different Config File"
SYNTAX	`Widget CreateCheckBox(Widget parent, GlobalPtr gp)`
	`Widget parent`	parent widget ID
	`GlobalPtr gp`	Global Comfrc pointer
INCLUDES	```#include "comfrc_header.h" #include "comfrc.h" #include "onlinefrc.h" #include "offlineASE.h" #include "mDB_Indx.h"```
DESCRIPTION	Creates the checkbox for "Using different Config File" on the FileDialog

FUNCTION	CreateFileComboBox	
PURPOSE	Create File Combo box for the File Dialog	
SYNTAX	void CreateFileComboBox(Widget parent, FileDialogComboBoxItem *cbox, Widget checkbox, int item_no, Boolean online, Boolean sensitive, GlobalPtr gp)	
	Widget parent	parent widget ID
	FileDialogComboBox Item *cbox	structure containing button and button callback information
	Widget checkbox	checkbox widget ID
	int item_no	combo box item in "cbox"
	Boolean online	Boolean variable true for online run
	Boolean sensitive	Boolean variable true for active
	GlobalPtr gp	Global Comfrc pointer
INCLUDES	#include "comfrc_header.h" #include "comfrc.h" #include "onlinefrc.h" #include "offlineASE.h" #include "mDB_Indx.h"	
DESCRIPTION	Creates a combo box with the list of appropriate filenames	

FUNCTION	DataFileCallback	
PURPOSE	Callback function for DataFile combo box selection	
SYNTAX	void DataFileCallback(Widget w, Widget checkbox, int num_buttons)	
	Widget w	parent widget ID
	Widget checkbox	checkbox widget ID
INCLUDES	#include "comfrc_header.h" #include "comfrc.h" #include "onlinefrc.h" #include "offlineASE.h" #include "mDB_Indx.h"	
DESCRIPTION	When a data file is selected from the combo box, the "use different config file" checkbox and the contents of the ConfigFile box are reset	

FUNCTION	GetFileList	
PURPOSE	Get a list of all the files from a given sub-directory	
SYNTAX	`int GetFileList(char *dirname, XmStringTable file_list)`	
	`char *dirname`	directory name
	`XmStringTable file_list`	List of filenames
INCLUDES	`#include "comfrc_header.h"` `#include "comfrc.h"` `#include "onlinefrc.h"` `#include "offlineASE.h"` `#include "mDB_Indx.h"`	
DESCRIPTION	Gets a list of all the files from a given sub-directory. Returns the number of files in the list	

FUNCTION	OpenFileCancel	
PURPOSE	Callback function for the "CANCEL" button in Open File Dialog	
SYNTAX	`void OpenFileCancel(Widget w, FileDialogCallbackStruct *fcs)`	
	`Widget w`	parent widget ID
	`FileDialogCallbac kStruct *fcs`	callback structure information
INCLUDES	`#include "comfrc_header.h"` `#include "comfrc.h"` `#include "onlinefrc.h"` `#include "offlineASE.h"` `#include "mDB_Indx.h"`	
DESCRIPTION	cancels dialog	

FUNCTION	OpenFileDialog	
PURPOSE	Establish an Open File Dialog for comfrc	
SYNTAX	`void OpenFileDialog(GlobalPtr gp)`	
	`GlobalPtr gp`	Global Comfrc pointer

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "offlineASE.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	Creates the OpenFile Dialog for Comfrc and waits for response. If OK is pressed: 1. A configuration file is read (function init called) 2. Datafile is set up if online test or read if offline test 3. Test schedule list is input to the text field on the Comfrc interface 4. If online, filedialog is closed and Setup of Actuators dialog is created 5. If offline, filedialog is closed and comfrc_offline is called	

FUNCTION	OpenFileHelp	
PURPOSE	Callback function for the "HELP " pushbutton in Open File Dialog	
SYNTAX	`void OpenFileHelp(Widget w, caddr_t client_data,` `XmAnyCallbackStruct *cbs)`	
	`Widget w;`	parent widget ID
	`caddr_t client_data;`	client data
	`XmAnyCallbackStruct *cbs;`	callback structure

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "offlineASE.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	Currently not implemented	

FUNCTION	OpenFileOK	
PURPOSE	Call back routine for the "OK" pushbutton in OpenFile Dialog	
SYNTAX	`void OpenFileOK(Widget w, FileDialogCallbackStruct *fcs)`	
	`Widget w`	parent widget ID
	`FileDialogCallbackStruct *fcs`	callback structure information

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "offlineASE.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	When OK is pressed on the file dialog box the following is performed: 1. Check to see if datafile exists. If yes, warning dialog is posted and user can append, overwrite or cancel. 2. Data file, Config file and test schedule file are copied to their respective text fields on the main Comfrc application interface 3. The application Monfrc is started (if online test) using the correct config file	

FUNCTION	ResetCallback	
PURPOSE	Callback function for Reset pushbutton on SetUpActuatorsDialog	
SYNTAX	`void ResetCallback(Widget w, int *reset, XmPushButtonCallbackStruct *cbs)`	
	`Widget w`	parent widget ID
	`int *finished`	return reset value of 2
	`XmPushButtonCallbackStruct *cbs`	callback structure information for cancel push button

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "offlineASE.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	Sets reset command to actuators and returns a value of 1 for reset variable.	

FUNCTION	SetConfigFileBox	
PURPOSE	Set the contents of the Config File Combo Box	
SYNTAX	`void SetConfigFileBox(Widget checkbox, Boolean new_file)`	
	`Widget checkbox`	checkbox widget ID
	`Boolean new_file`	Boolean value true if new file

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "offlineASE.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	Sets the contents of the config file combo box with the config filename contained in the datafile	

FUNCTION	SetUpActuatorsDialog	
PURPOSE	Creates a dialog to inform user of setup progress of actuators	
SYNTAX	`void SetUpActuatorsDialog(GlobalPtr gp)`	
	`GlobalPtr gp`	Global Comfrc pointer

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "offlineASE.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	Creates a dialog for the setup of actuators. When setup is complete, dialog will close. If error occur in setting up, error message will be displayed. User may then cancel or reset the actuators again.	

FUNCTION	trim_trailing_spaces	
PURPOSE	Trim off Trailing spaces from a given string	
SYNTAX	`void trim_trailing_spaces(char *line)`	
	`Char *line`	line of characters

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "onlinefrc.h"`
	`#include "offlineASE.h"`
	`#include "mDB_Indx.h"`	

DESCRIPTION	Trims the trailing spaces off a string and returns	

FUNCTION	Find_Fmt	
PURPOSE	Searches for `String` in file `fp`, then returns file pointer to original position	
SYNTAX	`int Find_Fmt(FILE *fp, const char *String)`	
	`FILE *fp`	Search File pointer
	`const char *String`	Search string

INCLUDES	`#include "initfrc.h"`
	`#include "debugs.h"`
	`#include "prtout.h"`
	`#include "display.h"`
	`#include "initModules.h"`	

DESCRIPTION	This function searches for `String` in file `fp`, then returns file pointer to original position. If string is found returns 1 else 0	

FUNCTION	FndNextLine	
PURPOSE	Reads file `fp` until the next valid line is reached.	
SYNTAX	`int FindNextLine(FILE *fp, char *Line)`	
	`FILE *fp`	File pointer
	`char *Line`	valid line

INCLUDES	`#include "initfrc.h"`
	`#include "debugs.h"`
	`#include "prtout.h"`
	`#include "display.h"`
	`#include "initModules.h"`	

DESCRIPTION	This function reads `fp` until the next valid line is reached. All lines beginning with % are skipped. If a valid line is found it returns 0, otherwise 1.	

FUNCTION	Find_Var	
PURPOSE	Finds a specified string within a specified file	
SYNTAX	`int Find_Var(FILE *ifp, char *String, char *Line)`	
	`FILE *ifp`	File pointer
	`char *String`	Search string
	`char *Line`	Line of characters

INCLUDES	`#include "initfrc.h"`
	`#include "debugs.h"`
	`#include "prtout.h"`
	`#include "display.h"`
	`#include "initModules.h"`	

DESCRIPTION	This function searches for `String` in the current `Line` and continues through the file until it is found (returns 0) or EOF is reached (returns 1).	

FUNCTION	init	
PURPOSE	Does initialisation for wind tunnel test	
SYNTAX	`int init(GlobalPtr gp, long *offset)`	
	`GlobalPtr gp`	Global Comfrc pointer
	`long *offset`	file position indicator

INCLUDES	`#include "initfrc.h"`
	`#include "debugs.h"`
	`#include "prtout.h"`
	`#include "display.h"`
	`#include "initModules.h"`	

DESCRIPTION	This function does all of the initialisation required for a wind tunnel test. It reads the contents of the configuration file and determines what data is to be acquired and stored in the data file. If online test, it outputs the header to the raw data file and reads in the test schedule file. If offline test, it checks the raw data file and configuration file for compatibility. Initialisation of the turntable and actuator modules is also performed. Initialisation of printout subroutines. Initialisation of block and line counters and various flags. If online test, initialisation of the Freestream module and inclinometer module	

FUNCTION	ReadCharArray	
PURPOSE	Splits a string (Line) into an array of sub-strings, which are separated by one or more spaces, or tabs.	
SYNTAX	int ReadCharArray(char *param[], char *Line)	
	char *param[]	Array of sub-strings
	char *Line	Search string
INCLUDES	#include "initfrc.h" #include "debugs.h" #include "prtout.h" #include "display.h" #include "initModules.h"	
DESCRIPTION	This function splits a string (Line) into an array of sub-strings, which are separated by one or more spaces, or tabs. It returns the number of sub-strings found.	

FUNCTION	Read_Config_File	
PURPOSE	Reads all configuration data from the supplied configuration file	
SYNTAX	`int Read_Config_File(ConfigDataPtr cdp, char* Filename)`	
	`ConfigDataPtr cdp`	Configuration Data pointer
	`char *Filename`	filename of the Configuration File

INCLUDES	`#include "initfrc.h"`
	`#include "debugs.h"`
	`#include "prtout.h"`
	`#include "display.h"`
	`#include "initModules.h"`	

DESCRIPTION	Opens the configuration file specified. If it is not a 'new format' configuration file it calls `Read_Old_Configfile`. The following configuration data is read and placed into `ConfigDataPtr`: 1. Title, customer, description lines 2. File/block constants/variables 3. Reference lengths and areas 4. Interference parameters 5. Power propeller required and propeller diameter 6. Mount type and source of attitude data 7. Balance offsets and sensor attitude offsets 8. Balance calibration matrices and units 9. Balance deflection matrices 10. Balance safeloads 11. Tareweight matrix 12. Whether sampling is required and the number of samples 13. Output format	

FUNCTION	Read_Old_Config_File	
PURPOSE	Reads all configuration data from the supplied configuration file (of the old format)	
SYNTAX	`int Read_Old_Config_File(ConfigDataPtr cdp, FILE *ifp)`	
	`ConfigDataPtr cdp`	Configuration Data pointer
	`FILE *ifp`	Configuration File pointer

INCLUDES	`#include "initfrc.h"`
	`#include "debugs.h"`
	`#include "prtout.h"`
	`#include "display.h"`
	`#include "initModules.h"`	

DESCRIPTION	This function reads all of the configuration data from an 'old format' configuration file into `ConfigDataPtr` (see `Read_Config_File`).	

FUNCTION	ReadSchedFile	
PURPOSE	Reads the test schedule information from the specified Test Schedule file into a `SchedFilePtr`	
SYNTAX	`int ReadSchedFile(SchedFilePtr sfp, char *SchedFile)`	
	`SchedFilePtr sfp`	Schedule File information pointer
	`char *SchedFile`	Schedule filename

INCLUDES	`#include "initfrc.h"`
	`#include "debugs.h"`
	`#include "prtout.h"`
	`#include "display.h"`
	`#include "initModules.h"`	

DESCRIPTION	This function reads the header from the specified test schedule file to obtain the parameters in the test schedule. It also reads the tolerance value for each of these parameters. It then reads each data line in the test schedule file and interprets it as a first/final zero, block end or a valid data line. If it is a valid data line it reads each of the parameter values into a linked list.	

FUNCTION	Read_Vector	
PURPOSE	Reads numbers from a string into a vector	
SYNTAX	`int Read_Vector(char *Line, double *x, int n)`	
	`char *Line`	string of numbers
	`double *x`	array
	`int n`	number of elements

INCLUDES	`#include "initfrc.h"`
	`#include "debugs.h"`
	`#include "prtout.h"`
	`#include "display.h"`
	`#include "initModules.h"`	

DESCRIPTION	This function searches a string for numbers separated by spaces or tabs and places the numbers into a vector of type `double`. Returns 0 upon successful completion, returns 1 otherwise.	

FUNCTION	Write_Config_File	
PURPOSE	Writes configuration data in the `ConfigDataPtr` to a file	
SYNTAX	`void Write_Config_File(ConfigDataPtr cdp, char *Filename)`	
	`ConfigDataPtr cdp`	Configuration Data pointer
	`char *Filename`	Configuration Filename

INCLUDES	`#include "initfrc.h"`
	`#include "debugs.h"`
	`#include "prtout.h"`
	`#include "display.h"`
	`#include "initModules.h"`	

DESCRIPTION	This function writes a configuration file with all of the configuration data currently stored in the `ConfigDataPtr`.	

FUNCTION	hostZero	
PURPOSE	Writes the inclinometer offsets contained in the modules initialisation file down to the Inclinometer Module	
SYNTAX	`void hostZero(char *inifile)`	
	`char *inifile`	modules initialisation file

INCLUDES	`#include "mDB_Indx.h"`
	`#include "moduleComms.h"`
	`#include "ReadFromIniFile.h"`
	`#include "initModules.h"`	

DESCRIPTION	This function writes the inclinometer offsets contained in the modules initialisation file down to the Inclinometer Module	

FUNCTION	initFs	
PURPOSE	Does initialisation of the Freestream Module	
SYNTAX	`void initFs(char *inifile, ConfigDataPtr cdp)`	
	`ConfigDataPtr cdp`	Configuration Data pointer
	`char *inifile`	modules initialisation file

INCLUDES	`#include "mDB_Indx.h"`
	`#include "moduleComms.h"`
	`#include "ReadFromIniFile.h"`
	`#include "initModules.h"`	

DESCRIPTION	This function sets the Reynolds number length from `ConfigDataPtr` to the Freestream Module	

FUNCTION	initInc	
PURPOSE	Does initialisation of the Inclinometer Module	
SYNTAX	`void initInc(char *inifile)`	
	`char *inifile`	modules initialisation file

INCLUDES	`#include "mDB_Indx.h"`
	`#include "moduleComms.h"`
	`#include "ReadFromIniFile.h"`
	`#include "initModules.h"`	

DESCRIPTION	This function does a `hostZero` of the Inclinometer Module. It sends down the inclinometer offsets obtained from the modules initialisation file to the Inclinometer Module.	

FUNCTION	initSCR
PURPOSE	Does initialisation of Sting Column Rig
SYNTAX	`void initSCR(char *inifile)`
	`char *inifile` — modules initialisation file

INCLUDES	`#include "mDB_Indx.h"`
	`#include "moduleComms.h"`
	`#include "ReadFromIniFile.h"`
	`#include "initModules.h"`

DESCRIPTION	This function reads the height offset of the Sting Column Rig from the modules initialisation file and writes it to the module.

FUNCTION	initTurntable
PURPOSE	Does initialisation of Turntable Module
SYNTAX	`void initTurntable(char *inifile, TurntableDataPtr tdp)`
	`char *inifile` — modules initialisation file
	`TurntableDataPtr` — Turntable Data pointer

INCLUDES	`#include "mDB_Indx.h"`
	`#include "moduleComms.h"`
	`#include "ReadFromIniFile.h"`
	`#include "initModules.h"`

DESCRIPTION	This function reads from the modules initialisation file, the valid turntables and their angle offsets and places this information in the `TurntableDataPtr`.

FUNCTION	selfZero
PURPOSE	Performs a "zero" of the inclinometer
SYNTAX	`void selfZero(char *inifile)`
	`char *inifile` — modules initialisation file

INCLUDES	`#include "mDB_Indx.h"`
	`#include "moduleComms.h"`
	`#include "ReadFromIniFile.h"`
	`#include "initModules.h"`

DESCRIPTION	This function reads the accelerometer angles from the Inclinometer Module and writes them as offsets to the modules initialisation file and down to the Inclinometer Module itself to "zero" the inclinometer.

FUNCTION	setReNoLength	
PURPOSE	Writes the Reynolds Number length down to the Freestream Module for correct calculation and display of Reynolds number.	
SYNTAX	`void setReNoLength(double renlen)`	
	`double renlen`	Reynolds Number length

INCLUDES	`#include "mDB_Indx.h"`
	`#include "moduleComms.h"`
	`#include "ReadFromIniFile.h"`
	`#include "initModules.h"`	

DESCRIPTION	This function writes the Reynolds Number length down to the Freestream Module for correct calculation and display of Reynolds number.	

FUNCTION	streamAllModules
PURPOSE	Sends a streaming command to all communicable modules
SYNTAX	`void streamAllModules()`

INCLUDES	`#include "mDB_Indx.h"`
	`#include "moduleComms.h"`
	`#include "ReadFromIniFile.h"`
	`#include "initModules.h"`

DESCRIPTION	This function sends a streaming command to all communicable modules.

FUNCTION	input_offline	
PURPOSE	Inputs data from a raw data file	
SYNTAX	`int input_offline(RawDataPtr rdp, LocalDataPtr ldp, CurrentLinePtr clp, long *offset)`	
	`RawDataPtr rdp`	Raw Data pointer
	`LocalDataPtr ldp`	Local Comfrc data pointer
	`CurrentLinePtr clp`	Current Line pointer
	`long *offset`	file position offset

INCLUDES	`#include "comfrc_header.h"`
	`#include "debugs.h"`
	`#include "DataAcqFuncs.h"`	

DESCRIPTION	This function opens the raw data file and reads in the header information and raw data lines into `RawDataPtr`.	

FUNCTION	input_online	
PURPOSE	Acquires a line of data from the instrumentation modules	
SYNTAX	`int input_online(GlobalPtr gp, int zero_line)`	
	`GlobalPtr gp`	Global Comfrc pointer
	`int zero_line`	=1 if a zero line

INCLUDES	`#include "comfrc_header.h"`
	`#include "debugs.h"`
	`#include "DataAcqFuncs.h"`	

DESCRIPTION	This function acquires a line of data from the instrumentation modules and places it into `RawDataPtr` according to raw data file header information defined in `init`.	

FUNCTION	CheckStatus	
PURPOSE	This is the timeout procedure for the moving dialog, which continually updates the current wind tunnel parameter values and the movement status indicators. It also indicates whether current values are within tolerance of the test schedule file setpoints.	
SYNTAX	`int CheckStatus(GlobalPtr gp)`	
	`GlobalPtr gp`	Global Comfrc pointer

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "mDB_Indx.h"`
	`#include "mdl_cntrl.h"`	

DESCRIPTION	This function checks the move status on the actuators and sting column rig. It also updates the current values of the wind tunnel parameters displayed on this screen. The background of the text boxes of the current values will continually be updated with one of the following colours: Green: NO movement AND the current value is within tolerance of the setpoint (if tolerances are applied) Orange: NO movement AND the current value is outside of the tolerance on the setpoint (if tolerances are applied) Red: MOVING The message box will be updated to report any actions taken and the current status.	

FUNCTION	close_cb	
PURPOSE	This is the Close button callback for the moving dialog	
SYNTAX	`void close_cb(Widget w, GlobalPtr gp)`	
	`Widget w`	parent widget ID
	`GlobalPtr gp`	Global Comfrc pointer

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "mDB_Indx.h"`
	`#include "mdl_cntrl.h"`	

DESCRIPTION	This function sets the flag `gp->sfp-msp->close_moving_dialog` to 1 which will cause the moving dialog to close.	

FUNCTION	move_cb	
PURPOSE	This is the Move button callback for the moving dialog	
SYNTAX	`void move_cb(Widget w, GlobalPtr gp)`	
	`Widget w`	parent widget ID
	`GlobalPtr gp`	Global Comfrc pointer

INCLUDES	`#include "comfrc_header.h"`
	`#include "comfrc.h"`
	`#include "mDB_Indx.h"`
	`#include "mdl_cntrl.h"`	

DESCRIPTION	This function obtains the setpoint values from the textboxes on the moving dialog and checks them for target angle errors. If no errors, a move command is sent to the actuators and/or Sting Column Rig.	

FUNCTION	move_online
PURPOSE	Creates a dialog on the Comfrc interface to show test schedule file parameter setpoints, current values and movement status.
SYNTAX	void move_online(Widget w, GlobalPtr gp)
	Widget w	parent widget ID
	GlobalPtr gp	Global Comfrc pointer
INCLUDES	#include "comfrc_header.h" #include "comfrc.h" #include "mDB_Indx.h" #include "mdl_cntrl.h"
DESCRIPTION	This function creates a dialog on the Comfrc interface to show test schedule file parameter setpoints, current values and the movement status. This dialog is displayed when a move command is initiated or when the manual setpoint button is pressed on the Comfrc interface. The buttons located on this window are Move, Stop, Close. A message area will inform the user of the current status. When a move has been initiated, any key press will abort the move. A time out procedure is implemented to update the current values and movement status indicators. If in semi or fully auto mode, the application will immediately attempt a move to the current test schedule setpoints when this window is displayed. When the move is complete and all parameters are within tolerance then the dialog will automatically close.

FUNCTION	stop_cb
PURPOSE	This is the Stop button callback for the moving dialog
SYNTAX	void stop_cb(Widget w, GlobalPtr gp)
	Widget w	parent widget ID
	GlobalPtr gp	Global Comfrc pointer
INCLUDES	#include "comfrc_header.h" #include "comfrc.h" #include "mDB_Indx.h" #include "mdl_cntrl.h"
DESCRIPTION	This function sends a stop command to the actuators and/or the Sting Column Rig to abort a move.

FUNCTION	wprint	
PURPOSE	Writes formatted output to text widgets	
SYNTAX	`void wprint(va_alist)`	
	`va_alist`	variable argument list

INCLUDES	`#include <varargs.h`
	`#include <ctype.h`
	`#include <Xm/Xm.h`
	`#include <Xm/Text.h`	

DESCRIPTION	This function allows the programmer to write formatted output to an X text widget. The function takes the following form: wprint (X Text widget, " formatted text ", variables)	

FUNCTION	average	
PURPOSE	Finds the average of the first and final zeros and calculates the buoyant zeros.	
SYNTAX	`int average(ConfigDataPtr cdp, LocalDataPtr ldp,` `ZeroDataPtr zdp)`	
	`ConfigDataPtr cdp`	Configuration Data pointer
	`LocalDataPtr ldp`	Local Comfrc data pointer
	`ZeroDataPtr zdp`	Zero Data pointer
INCLUDES	`#include "normalfrc.h"` `#include "debugs.h"`	
DESCRIPTION	This function finds the average of the first and final zeros and calculates the buoyant zeros.	

FUNCTION	chekit	
PURPOSE	Checks zero lines for consistency of attitude.	
SYNTAX	`int chekit(GlobalPtr gp)`	
	`GlobalPtr gp`	Global Comfrc pointer
INCLUDES	`#include "normalfrc.h"` `#include "debugs.h"`	
DESCRIPTION	This function checks zero lines for consistency of attitude and is only called if the current line has been identified as a zero line. If it is a first zero line then the zero values are assigned to the zero data If it is a final zero then the first and final zero strain gauge values are checked for consistency. If outside a tolerance of 0.02V a warning is issued.	

FUNCTION	convert_data	
PURPOSE	Converts one raw data line to engineering units	
SYNTAX	`int convert_data(GlobalPtr gp, CurrentLinePtr clp)`	
	`GlobalPtr gp`	Global Comfrc pointer
	`CurrentLinePtr clp`	Current Line pointer
INCLUDES	`#include "normalfrc.h"` `#include "debugs.h"`	
DESCRIPTION	This function converts one raw data line to engineering units	

FUNCTION	FindFileConstIndex	
PURPOSE	Finds whether the specified file constant exists in the `ConfigDataPtr`	
SYNTAX	`int FindFileConstIndex(ConfigDataPtr cdp, char *fconst)`	
	`ConfigDataPtr cdp`	Configuration Data pointer
	`char *fconst`	file constant string
INCLUDES	`#include "normalfrc.h"` `#include "debugs.h"`	
DESCRIPTION	This function finds whether the specified file constant exists in the `ConfigDataPtr`	

FUNCTION	hextodec	
PURPOSE	Converts 2 word parameters to the correct integer format	
SYNTAX	`double hextodec(int upper, int lower)`	
	`int upper`	upper word
	`int lower`	lower word
INCLUDES	`#include "normalfrc.h"` `#include "debugs.h"`	
DESCRIPTION	This function converts 2 word parameters to the correct integer format	

FUNCTION	normal	
PURPOSE	Processes a normal data line	
SYNTAX	`int normal(GlobalPtr gp)`	
	`GlobalPtr gp`	Global Comfrc pointer
INCLUDES	`#include "normalfrc.h"` `#include "debugs.h"`	
DESCRIPTION	This function processes a normal raw data line to engineering units and coefficients. Firstly, if sampling is not being done, the data is converted to engineering units. If the data line is a first or final zero, the buoyant zeros are calculated. If it is a final zero, all data is then reprocessed through `calall` based on the new buoyant zeros. If the data line is a normal force data line then the strain gauge voltages are reduced to coefficients through calls to the functions `balred`, `tarewt`, `attmod`, `towind`, `toaxes`. The data is also then displayed on the `dspfrc` application.	

FUNCTION	printout	
PURPOSE	Outputs to the screen the current line values computed from the raw data values	
SYNTAX	`int printout(RawDataPtr rdp, CurrentLinePtr clp)`	
	`RawDataPtr rdp`	Raw Data pointer
	`CurrentLinePtr clp`	Current Line pointer
INCLUDES	`#include "normalfrc.h"` `#include "debugs.h"`	
DESCRIPTION	This function outputs to the screen the current line values computed from the raw data values	

FUNCTION	sgdata	
PURPOSE	Converts raw strain gauge data to a voltage	
SYNTAX	`double sgdata(int nData)`	
	`int nData`	raw strain gauge data
INCLUDES	`#include "normalfrc.h"` `#include "debugs.h"`	
DESCRIPTION	This function converts raw strain gauge data to a voltage	

FUNCTION	sortit	
PURPOSE	Determines whether a line is a zero line or a normal force data line	
SYNTAX	`int sortit(CurrentLinePtr clp)`	
	`CurrentLinePtr clp`	Current Line pointer
INCLUDES	`#include "normalfrc.h"` `#include "debugs.h"`	
DESCRIPTION	This function determines whether a line is a zero line or a normal force data line by checking the airspeed is above or below a value of 5m/s	

FUNCTION	CalculateX1invX2	
PURPOSE	Calculates the X1inv matrix and performs the matrix multiplication X1inv * X2	
SYNTAX	`int CalculateX1invX2(GlobalPtr gp)`	
	`GlobalPtr gp`	Global Comfrc data pointer

INCLUDES	`#include "initfrc.h"`
	`#include "onlinefrc.h"`
	`#include "comfrc_header.h"`
	`#include "nrutil.h"`
	`#include "prtout.h"`	

DESCRIPTION	This function calculates the X1 inv matrix and performs the matrix multiplication X1inv * X2	

FUNCTION	ConvertDeflConstFormat	
PURPOSE	Converts the deflection matrix read in from the ASE2000 TWT data file to the LSWT format	
SYNTAX	`int ConvertDeflConstFormat(MatrixPtr defl_temp,` `GlobalPtr gp)`	
	`MatrixPtr defl_temp`	Matrix pointer
	`GlobalPtr gp`	Global Comfrc pointer

INCLUDES	`#include "initfrc.h"`
	`#include "onlinefrc.h"`
	`#include "comfrc_header.h"`
	`#include "nrutil.h"`
	`#include "prtout.h"`	

DESCRIPTION	This function converts the deflection matrix (`defl_temp`) read in from the ASE2000 TWT data file to the `ConfigDataPtr` in `GlobalPtr` (LSWT format)	

FUNCTION	ConvertDMatrixAxesFormat	
PURPOSE	Converts the D matrix read in from the ASE2000 TWT data file (X,Y,Z,l,m,n format) to the LSWT format (X,l,Y,m,Z,n)	
SYNTAX	`int ConvertDMatrixAxesFormat(MatrixPtr dtemp, GlobalPtr gp)`	
	`MatrixPtr dtemp`	Matrix pointer
	`GlobalPtr gp`	Global Comfrc pointer

INCLUDES	`#include "initfrc.h"`
	`#include "onlinefrc.h"`
	`#include "comfrc_header.h"`
	`#include "nrutil.h"`
	`#include "prtout.h"`	

DESCRIPTION	This function converts the D matrix (`dtemp`) read in from the ASE2000 TWT data file (X,Y,Z,l,m,n format) to the `ConfigDataPtr` in `GlobalPtr` (LSWT format X,l,Y,m,Z,n)	

FUNCTION	ConvertTaresFormat	
PURPOSE	Converts the tares matrix read in from the ASE2000 TWT data file to the LSWT format	
SYNTAX	`int ConvertTaresFormat(MatrixPtr tares_temp, GlobalPtr gp)`	
	`MatrixPtr tares_temp`	Matrix pointer
	`GlobalPtr gp`	Global Comfrc pointer

INCLUDES	`#include "initfrc.h"`
	`#include "onlinefrc.h"`
	`#include "comfrc_header.h"`
	`#include "nrutil.h"`
	`#include "prtout.h"`	

DESCRIPTION	This function converts the tares matrix (`tares_temp`) read in from the ASE2000 TWT data file to the `ConfigDataPtr` in `GlobalPtr` (LSWT format)	

81

FUNCTION	ConvertX1MatrixAxesFormat	
PURPOSE	Converts the X1 matrix read in from the ASE2000 TWT data file (X,Y,Z,l,m,n format) to the LSWT format (X,l,Y,m,Z,n)	
SYNTAX	`int ConvertX1MatrixAxesFormat(MatrixPtr x1temp, GlobalPtr gp)`	
	`MatrixPtr x1temp`	Matrix pointer
	`GlobalPtr gp`	Global Comfrc pointer

INCLUDES	`#include "initfrc.h"`
	`#include "onlinefrc.h"`
	`#include "comfrc_header.h"`
	`#include "nrutil.h"`
	`#include "prtout.h"`	

DESCRIPTION	This function converts the X1 matrix (x1temp) read in from the ASE2000 TWT data file (X,Y,Z,l,m,n format) to the `ConfigDataPtr` in `GlobalPtr` (LSWT format X,l,Y,m,Z,n)	

FUNCTION	ConvertX2MatrixAxesFormat	
PURPOSE	Converts the X2matrix read in from the ASE2000 TWT data file (X,Y,Z,l,m,n format) to the LSWT format (X,l,Y,m,Z,n)	
SYNTAX	`int ConvertX2MatrixAxesFormat(MatrixPtr x2temp, GlobalPtr gp)`	
	`MatrixPtr x2temp`	Matrix pointer
	`GlobalPtr gp`	Global Comfrc pointer

INCLUDES	`#include "initfrc.h"`
	`#include "onlinefrc.h"`
	`#include "comfrc_header.h"`
	`#include "nrutil.h"`
	`#include "prtout.h"`	

DESCRIPTION	This function converts the X2 matrix (x2temp) read in from the ASE2000 TWT data file (X,Y,Z,l,m,n format) to the `ConfigDataPtr` in `GlobalPtr` (LSWT format X,l,Y,m,Z,n)	

FUNCTION	offlineASE	
PURPOSE	Processes ASE2000 transonic wind tunnel data offline	
SYNTAX	`int offlineASE(GlobalPtr gp)`	
	`GlobalPtr gp`	Global Comfrc pointer

INCLUDES	`#include "initfrc.h"`
	`#include "onlinefrc.h"`
	`#include "comfrc_header.h"`
	`#include "nrutil.h"`
	`#include "prtout.h"`	

DESCRIPTION	This function processes ASE2000 transonic wind tunnel data offline. It calls `Read_Config_ASE` to read the configuration data and then calls either `offlineASEfrc` or `offlineASEtwt` depending on whether it is a force and moment test or a tareweight calculation.	

FUNCTION	offlineASEfrc	
PURPOSE	Reads and processes ASE2000 TWT force and moment data files offline	
SYNTAX	`int offlineASEfrc(GlobalPtr gp)`	
	`GlobalPtr gp`	Global Comfrc pointer

INCLUDES	`#include "initfrc.h"`
	`#include "onlinefrc.h"`
	`#include "comfrc_header.h"`
	`#include "nrutil.h"`
	`#include "prtout.h"`	

DESCRIPTION	This function reads ASE2000 TWT force and moment data files and then processes all data to coefficient form through the Comfrc routines. 1. Reads first and final zero strain gauge information from ASE2000 TWT data file and sets up zero data pointers. 2. Reads the header and identifies which column corresponds to which parameter. 3. Reads all data into `CurrentLinePtr` according to the sampling characteristics defined in the LSWT configuration file. Calculation and output of standard deviations if required. 4. Reads buoyant pitch, roll yaw attitudes. 5. Calculates buoyant zeros using `average`. 6. Calls `calall` to process all data and output results according to output format defined in the LSWT configuration file.	

FUNCTION	offlineASEtwt
PURPOSE	Reads and processes ASE2000 TWT tareweight data offline
SYNTAX	`int offlineASEtwt(GlobalPtr gp)`
	`GlobalPtr gp` — Global Comfrc pointer

INCLUDES	`#include "initfrc.h"`
	`#include "onlinefrc.h"`
	`#include "comfrc_header.h"`
	`#include "nrutil.h"`
	`#include "prtout.h"`

DESCRIPTION	This function reads ASE2000 TWT tareweight data and then processes all data to obtain the tareweight matrix through the Comfrc routines. 1. Reads first and final zero strain gauge information from a data file called `ASEtwt.dat` that contains pitch and roll attitudes with all 6 strain gauge values. 2. Calls `average` to calculate buoyant zeros. 3. Calls `prtini_twt` to print the header of the tareweight output file 4. Calls `calalltwt` to process all data using the calibration matrices obtained from a specified ASE2000 TWT data file, and outputs the calculated tareweight matrix.

FUNCTION	Read_Config_ASE
PURPOSE	Reads in the configuration data from both the ASE2000 transonic wind tunnel data file and a LSWT configuration file
SYNTAX	`int Read_Config_ASE(GlobalPtr gp)`
	`GlobalPtr gp` — Global Comfrc pointer

INCLUDES	`#include "initfrc.h"`
	`#include "onlinefrc.h"`
	`#include "comfrc_header.h"`
	`#include "nrutil.h"`
	`#include "prtout.h"`

DESCRIPTION	This function reads in all configuration data required from the ASE2000 transonic wind tunnel data file. Also reads in the output format, sampling characteristics, wind tunnel corrections required and base pressure required from a LSWT configuration file

FUNCTION	ReadParameterMatrix	
PURPOSE	Searches for the header in the ASE2000 TWT data file and then reads in the matrix on the following lines according to the number of rows and columns specified	
SYNTAX	`int ReadParameterMatrix(FILE *ifp, char *header, MatrixPtr mp, int nrows, int ncols)`	
	`FILE *ifp`	Data File pointer
	`char *header`	Search header
	`MatrixPtr mp`	general matrix pointer containing all elements
	`int nrows`	number of rows in the matrix
	`int ncols`	number of columns in the matrix

INCLUDES	`#include "initfrc.h"`
	`#include "onlinefrc.h"`
	`#include "comfrc_header.h"`
	`#include "nrutil.h"`
	`#include "prtout.h"`	

DESCRIPTION	This function searches for the header in the ASE2000 TWT data file and then reads in the matrix on the following lines according to the number of rows and columns specified	

FUNCTION	ReadParameterString	
PURPOSE	Searches for the header in the ASE2000 TWT data file and then reads in the string on the following line	
SYNTAX	`int ReadParameterString(FILE *ifp, char *header, char *string)`	
	`FILE *ifp`	Data File pointer
	`char *header`	Search header
	`char *string`	returned string on the line following the search header

INCLUDES	`#include "initfrc.h"`
	`#include "onlinefrc.h"`
	`#include "comfrc_header.h"`
	`#include "nrutil.h"`
	`#include "prtout.h"`	

DESCRIPTION	This function searches for the header in the ASE2000 TWT data file and then reads in the string on the following line.	

FUNCTION	ReadParameterValue	
PURPOSE	Searches for the header in the ASE2000 TWT data file and then reads in the value on the following line	
SYNTAX	int ReadParameterValue(FILE *ifp, char *header, double *value)	
	FILE *ifp	Data File pointer
	char *header	Search header
	double *value	returned value on the line following the search header
INCLUDES	#include "initfrc.h" #include "onlinefrc.h" #include "comfrc_header.h" #include "nrutil.h" #include "prtout.h"	
DESCRIPTION	This function searches for the header in the ASE2000 TWT data file and then reads in the value on the following line.	

FUNCTION	comfrc_offline	
PURPOSE	Performs off-line computation of raw data for Comfrc	
SYNTAX	comfrc_offline(Widget w, GlobalPtr gp)	
	Widget w	parent widget ID
	GlobalPtr gp	Global Comfrc pointer
INCLUDES	#include "offlinefrc.h" #include "onlinefrc.h" #include "comfrc_header.h" #include "normalfrc.h" #include "debugs.h"	
DESCRIPTION	This function performs off-line computation of raw data for Comfrc. Calls input_offline to get a line of raw data from the file. If sampling is specified then it calls sample_data_offline otherwise it calls normal to process the data line. Output is put to the text widget on the main Comfrc interface. When the first and final zeros have been read in then the function calls the appropriate calall to process all data using the first and final zero data.	

FUNCTION	read_dataline_offline	
PURPOSE	Reads a line of raw data knowing that it is only data. Called in the offline sampling case	
SYNTAX	read_dataline_offline(RawDataPtr rdp, LocalDataPtr ldp, long *offset)	
	RawDataPtr rdp	Raw Data pointer
	LocalDataPtr ldp	Local Comfrc data pointer
	long *offset	file position locater in raw data file
INCLUDES	#include "offlinefrc.h" #include "onlinefrc.h" #include "comfrc_header.h" #include "normalfrc.h" #include "debugs.h"	
DESCRIPTION	This function reads a line of raw data knowing that it is only data. Called in the offline sampling case	

FUNCTION	sample_data_offline	
PURPOSE	Performs the processing of the data in the off-line case if sampling is required. It includes statistical computations.	
SYNTAX	int sample_data_offline(GlobalPtr gp, long *offset)	
	GlobalPtr gp	Global Comfrc pointer
	long *offset	file pointer location in raw data file
INCLUDES	#include "offlinefrc.h" #include "onlinefrc.h" #include "comfrc_header.h" #include "normalfrc.h" #include "debugs.h"	
DESCRIPTION	This function performs the processing of the data in the off-line case if sampling is required. For each sample it reads in a raw data line, converts to engineering units and cumulates the statistical information. It then calls normal to process the average of the samples to coefficient form and outputs standard deviations to a statistical file.	

FUNCTION	add_to_list
PURPOSE	Allocates memory for another pointer after the last in the linked list and moves to that pointer
SYNTAX	`CurrentLinePtr add_to_list(CurrentLinePtr last)`
	`CurrentLinePtr last` last pointer in linked list

INCLUDES	`#include "onlinefrc.h"`
	`#include "comfrc_header.h"`
	`#include "normalfrc.h"`
	`#include "schedule.h"`
	`#include "debugs.h"`

DESCRIPTION	This function allocates memory for another pointer after the last in the linked list and moves to that pointer

FUNCTION	calculate_stats	
PURPOSE	Calculates the statistical information for the samples at each data point	
SYNTAX	`int calculate_stats(GlobalPtr gp, CurrentLinePtr sigmax, CurrentLinePtr sigmaxx, CurrentLinePtr stdevclp, double n)`	
	`GlobalPtr gp`	Global Comfrc pointer
	`CurrentLinePtr sigmax`	Current Line pointer for sum of data samples
	`CurrentLinePtr sigmaxx`	Current Line pointer for sum of the squares of the data samples
	`CurrentLinePtr stdevclp`	Current Line pointer for the standard deviation of the data samples
	`double n`	number of samples

INCLUDES	`#include "onlinefrc.h"`
	`#include "comfrc_header.h"`
	`#include "normalfrc.h"`
	`#include "schedule.h"`
	`#include "debugs.h"`	

DESCRIPTION	This function calculates the statistical information for the samples at each data point. Using the sums of the parameters over every sample and the sums of the squares of the parameters over every sample, the standard deviation is calculated for that group of samples.	

FUNCTION	comfrc_online	
PURPOSE	Performs on-line computation of raw data for Comfrc	
SYNTAX	`comfrc_online(GlobalPtr gp)`	
	`GlobalPtr gp`	Global Comfrc pointer

INCLUDES	`#include "onlinefrc.h"`
	`#include "comfrc_header.h"`
	`#include "normalfrc.h"`
	`#include "schedule.h"`
	`#include "debugs.h"`	

DESCRIPTION	This function simply interpret what command has been received from the Comfrc interface during an on-line test and calls the appropriate on-line function:
	`get_data_online`
	`firstz_online`
	`finalz_online`
	`reject_online`
	`block_online`
	`next_online`
	`previous_online`
	`history_online`
	`help_online`	

FUNCTION	count	
PURPOSE	Returns the number of structures in the list pointed to by head	
SYNTAX	`int count(CurrentLinePtr head)`	
	`CurrentLinePtr head`	head of the linked list

INCLUDES	`#include "onlinefrc.h"`
	`#include "comfrc_header.h"`
	`#include "normalfrc.h"`
	`#include "schedule.h"`
	`#include "debugs.h"`	

DESCRIPTION	This function returns the number of structures in the list pointed to by head	

FUNCTION	cumulate_stats	
PURPOSE	Cumulates the statistical information for the samples at each data point	
SYNTAX	int cumulate_stats(GlobalPtr gp, CurrentLinePtr tempclp, CurrentLinePtr sigmax, CurrentLinePtr sigmaxx)	
	GlobalPtr gp	Global Comfrc pointer
	CurrentLinePtr tempclp	Current DataLine pointer
	CurrentLinePtr sigmax	Current Line pointer for sum of data samples
	CurrentLinePtr sigmaxx	Current Line pointer for sum of (data samples) squares
INCLUDES	#include "onlinefrc.h" #include "comfrc_header.h" #include "normalfrc.h" #include "schedule.h" #include "debugs.h"	
DESCRIPTION	This function cumulates the statistical information for the samples at each data point. Sums the parameters over every sample in the data point. Sums the squares of the parameters over every sample in the data point.	

FUNCTION	finalz_online	
PURPOSE	Acquires a final zero data line, sets the corresponding flag and reprocesses all data based on the first and final zero data	
SYNTAX	`finalz_online(Widget w, GlobalPtr gp)`	
	`Widget w`	parent widget ID
	`GlobalPtr gp`	Global Comfrc pointer

INCLUDES	`#include "onlinefrc.h"`
	`#include "comfrc_header.h"`
	`#include "normalfrc.h"`
	`#include "schedule.h"`
	`#include "debugs.h"`	

DESCRIPTION	This function acquires a final zero data line in the same manner as in `get_data_online`. It also sets the local final zero flags to TRUE to ensure the function `normal` interprets it as a zero line. It recalculates all data based on the first and final zero data by calling `calall`. The sensitivity of the first zero button on the Comfrc interface is turned on and the sensitivity of the other buttons are turned off.	

FUNCTION	find_penultimate	
PURPOSE	Returns the offset (file position) of the penultimate line of a file	
SYNTAX	`long find_penultimate(char *filename)`	
	`char *filename`	filename

INCLUDES	`#include "onlinefrc.h"`
	`#include "comfrc_header.h"`
	`#include "normalfrc.h"`
	`#include "schedule.h"`
	`#include "debugs.h"`	

DESCRIPTION	This function returns the offset (file position) of the penultimate line of a file.	

FUNCTION	firstz_online	
PURPOSE	Acquires a first zero data line and sets the corresponding flag	
SYNTAX	firstz_online(Widget w, GlobalPtr gp)	
	Widget w	parent widget ID
	GlobalPtr gp	Global Comfrc pointer
INCLUDES	#include "onlinefrc.h" #include "comfrc_header.h" #include "normalfrc.h" #include "schedule.h" #include "debugs.h"	
DESCRIPTION	This function acquires a first zero data line in the same manner as in get_data_online. It also sets the local first zero flags to TRUE to ensure the function normal interprets it as a zero line. The sensitivity of the first zero button on the Comfrc interface is turned off and the sensitivity of the other buttons are turned on.	

FUNCTION	get_data_online	
PURPOSE	Acquires data from the instrumentation modules, processes it and writes the raw data to a file	
SYNTAX	get_data_online(Widget w, GlobalPtr gp)	
	Widget w	parent widget ID
	GlobalPtr gp	Global Comfrc pointer
INCLUDES	#include "onlinefrc.h" #include "comfrc_header.h" #include "normalfrc.h" #include "schedule.h" #include "debugs.h"	
DESCRIPTION	This function acquires data from the instrumentation modules, processes it and writes the raw data to a file. If it is using the test schedule file in semi or fully auto mode then it will first try to move actuators/attitudes to their setpoints. It will also go to the next setpoint once data has been acquired. If tolerances are applied, the data that is acquired is checked against the setpoints. If any parameter lies outside its tolerance, the user is prompted to accept or reject the acquired data.	

FUNCTION	help_online	
PURPOSE	Brings up available help documentation (currently not implemented)	
SYNTAX	help_online(Widget w, GlobalPtr gp)	
	Widget w	parent widget ID
	GlobalPtr gp	Global Comfrc pointer
INCLUDES	#include "onlinefrc.h" #include "comfrc_header.h" #include "normalfrc.h" #include "schedule.h" #include "debugs.h"	
DESCRIPTION	This function brings up available help documentation (currently not implemented)	

FUNCTION	history_online	
PURPOSE	Performs history plot on dspfrc (currently not implemented)	
SYNTAX	history_online(Widget w, GlobalPtr gp)	
	Widget w	parent widget ID
	GlobalPtr gp	Global Comfrc pointer
INCLUDES	#include "onlinefrc.h" #include "comfrc_header.h" #include "normalfrc.h" #include "schedule.h" #include "debugs.h"	
DESCRIPTION	This function performs history plot on dspfrc (currently not implemented)	

FUNCTION	init_block	
PURPOSE	Initialises a new block in Comfrc	
SYNTAX	int init_block(LocalDataPtr ldp)	
	LocalDataPtr ldp	Local Comfrc data pointer
INCLUDES	#include "onlinefrc.h" #include "comfrc_header.h" #include "normalfrc.h" #include "schedule.h" #include "debugs.h"	
DESCRIPTION	This function initialises a new block in Comfrc and resets flags.	

FUNCTION	next_online	
PURPOSE	Prepares for the next setpoint in the test schedule file	
SYNTAX	next_online(Widget w, GlobalPtr gp)	
	Widget w	parent widget ID
	GlobalPtr gp	Global Comfrc pointer
INCLUDES	#include "onlinefrc.h" #include "comfrc_header.h" #include "normalfrc.h" #include "schedule.h" #include "debugs.h"	
DESCRIPTION	This function prepares for the next setpoint in the test schedule file by highlighting it in the test schedule field on the Comfrc interface and incrementing the test schedule file linked list pointer. If in fully auto mode, it will interpret the next setpoint and take action automatically whether it is to do a block_online or a get_data_online, including moving actuators/attitudes.	

FUNCTION	normaldata_to_widget	
PURPOSE	Outputs results from normal to a text widget on the Comfrc application interface	
SYNTAX	normaldata_to_widget(Widget w, GlobalPtr gp)	
	Widget w	parent widget ID
	GlobalPtr gp	Global Comfrc pointer
INCLUDES	#include "onlinefrc.h" #include "comfrc_header.h" #include "normalfrc.h" #include "schedule.h" #include "debugs.h"	
DESCRIPTION	This function outputs results from normal to a text widget (gp->xwp->data_text) on the Comfrc application interface	

FUNCTION	prev_online	
PURPOSE	Prepares for the previous setpoint in the test schedule file	
SYNTAX	prev_online(Widget w, GlobalPtr gp)	
	Widget w	parent widget ID
	GlobalPtr gp	Global Comfrc pointer
INCLUDES	#include "onlinefrc.h" #include "comfrc_header.h" #include "normalfrc.h" #include "schedule.h" #include "debugs.h"	
DESCRIPTION	This function prepares for the previous setpoint in the test schedule file by highlighting it in the test schedule field on the Comfrc interface and decrementing the test schedule file linked list pointer. If in fully auto mode, it will interpret the previous setpoint and take action automatically whether it is to do a block_online or a get_data_online, including moving actuators/attitudes.	

FUNCTION	reject_online	
PURPOSE	Rejects the last line of data that was acquired	
SYNTAX	reject_online(Widget w, GlobalPtr gp)	
	Widget w	parent widget ID
	GlobalPtr gp	Global Comfrc pointer
INCLUDES	#include "onlinefrc.h" #include "comfrc_header.h" #include "normalfrc.h" #include "schedule.h" #include "debugs.h"	
DESCRIPTION	This function rejects the last line of data that was acquired. The data line is deleted from the linked list and the raw data file is rewritten without the last line of data. If sampling was required, it removes all samples and also rewrites the statistical file without the last line of data. A first or final zero or block end cannot be rejected.	

FUNCTION	sample_data_online	
PURPOSE	Performs the sampling and processing of the data in the on-line case. It includes statistical computations.	
SYNTAX	int sample_data_online(GlobalPtr gp, int zero_line)	
	GlobalPtr gp	Global Comfrc pointer
	int zero_line	=1 if a zero line
INCLUDES	#include "onlinefrc.h" #include "comfrc_header.h" #include "normalfrc.h" #include "schedule.h" #include "debugs.h"	
DESCRIPTION	This function performs the sampling and processing of the data in the on-line case. It displays a dialog on the Comfrc interface to show the progress of the acquisition of each sample. For each sample it acquires a raw data line, writes it to the raw data file, converts to engineering units and cumulates the statistical information. It then calls normal to process the average of the samples to coefficient form and outputs standard deviations to a statistical file.	

FUNCTION	stdev	
PURPOSE	Calculates the standard deviation from the sum, the sum of the squares and the number of samples	
SYNTAX	double stdev(double sigmax, double sigmaxx, double n)	
	double sigmax	sum of data samples
	double sigmaxx	sum of the squares of the data samples
	double n	number of samples
INCLUDES	#include "onlinefrc.h" #include "comfrc_header.h" #include "normalfrc.h" #include "schedule.h" #include "debugs.h"	
DESCRIPTION	This function calculates the standard deviation from the sum, the sum of the squares and the number of samples. It returns the standard deviation.	

FUNCTION	string_to_widget
PURPOSE	Outputs a string to a text widget on the Comfrc application interface
SYNTAX	`string_to_widget(Widget w, GlobalPtr gp, char *s)`
	`Widget w` — parent widget ID
	`GlobalPtr gp` — Global Comfrc pointer
	`char *s` — string to be output

INCLUDES	`#include "onlinefrc.h"`
	`#include "comfrc_header.h"`
	`#include "normalfrc.h"`
	`#include "schedule.h"`
	`#include "debugs.h"`

DESCRIPTION	This function outputs a string to a text widget (`gp-xwp-data_text`) on the Comfrc application interface

FUNCTION	TakeDataSamples
PURPOSE	Obtains data samples and displays a working dialog on the Comfrc interface to indicate the progress of data acquisition

SYNTAX	`void TakeDataSamples(GlobalPtr gp, int zero_line,`
	`CurrentLinePtr tempclp, CurrentLinePtr sigmax,`
	`CurrentLinePtr sigmaxx)`

	`GlobalPtr gp` — Global Comfrc pointer
	`int zero_line` — =1 if zero line
	`CurrentLinePtr tempclp` — Current Data Line pointer
	`CurrentLinePtr sigmax` — Current Line pointer for sum of data samples
	`CurrentLinePtr sigmaxx` — Current Line pointer for sum of the squares of the data samples

INCLUDES	`#include "onlinefrc.h"`
	`#include "comfrc_header.h"`
	`#include "normalfrc.h"`
	`#include "schedule.h"`
	`#include "debugs.h"`

DESCRIPTION	This function displays a working dialog on the Comfrc interface to show the progress of acquisition of data samples. Each sample of raw data is acquired, written to the file, converted to engineering units, and statistical information is accumulated. Dialog is closed when the required number of samples has been acquired.

FUNCTION	write_rawdata_line	
PURPOSE	Writes a single line of acquired raw data to the raw data file	
SYNTAX	write_rawdata_line(GlobalPtr gp, long *offset)	
	GlobalPtr gp	Global Comfrc pointer
	long *offset	file pointer location in raw data file
INCLUDES	#include "onlinefrc.h" #include "comfrc_header.h" #include "normalfrc.h" #include "schedule.h" #include "debugs.h"	
DESCRIPTION	This function writes a single line of acquired raw data to the raw data file	

FUNCTION	prtini	
PURPOSE	Prints the header information to the results output file and statistics file (if sampling required) for a force and moment test	
SYNTAX	int prtini(GlobalPtr gp)	
	GlobalPtr gp	Global Comfrc pointer
INCLUDES	#include "prtout.h"	
DESCRIPTION	This function prints the header information to the results output file and statistics file (if sampling required) for a force and moment test.	

FUNCTION	prtini_twt	
PURPOSE	Prints the header information to the results output file and statistics file (if sampling required) for a tareweight calculation	
SYNTAX	int prtini_twt(ConfigDataPtr cdp, LocalDataPtr ldp)	
	ConfigDataPtr cdp	Configuration Data pointer
	LocalDataPtr ldp	Local Comfrc data pointer
INCLUDES	#include "prtout.h"	
DESCRIPTION	This function prints the header information to the results output file and statistics file (if sampling required) for a tareweight calculation.	

98

FUNCTION	prtblk	
PURPOSE	Outputs a blank line to the output results file and statistics file (if sampling required) to indicate a new block	
SYNTAX	`int prtblk(LocalDataPtr ldp, int sampling, int from_block_online)`	
	`LocalDataPtr ldp`	Local Comfrc data pointer
	`int sampling`	=1 if sampling required
	`int from_block_online`	=1 if called from `block_online`
INCLUDES	`#include "prtout.h"`	
DESCRIPTION	This function outputs a blank line to the output results file and statistics file (if sampling required) to indicate a new block	

FUNCTION	prtline	
PURPOSE	Outputs a line of processed data to the output results file according to the output format defined in the `ConfigDataPtr`.	
SYNTAX	`int prtline(CurrentLinePtr clp, GlobalPtr gp, char *outfile)`	
	`CurrentLinePtr clp`	Current Data Line pointer
	`GlobalPtr gp`	Global Comfrc pointer
	`char *outfile`	output file
INCLUDES	`#include "prtout.h"`	
DESCRIPTION	This function outputs a line of processed data to the output results file according to the output format defined in the `ConfigDataPtr`.	

FUNCTION	prtstdev	
PURPOSE	Outputs a line of standard deviations of certain parameters for each group of samples	
SYNTAX	`int prtstdev(CurrentLinePtr stdevclp, char *outfile)`	
	`CurrentLinePtr stdevclp`	Current Line pointer for the standard deviation of each group of samples
	`char *outfile`	output file
INCLUDES	`#include "prtout.h"`	
DESCRIPTION	This function outputs a line of standard deviations of certain parameters for each group of samples.	

FUNCTION	AddPaddingSpaces	
PURPOSE	Adds padding spaces to space out the header items above the test schedule field	
SYNTAX	void AddPaddingSpaces(char *output_line, char *line_buf)	
	char *output_line	final string of spaced header items
	char *line_buf	original string of header items
INCLUDES	#include "comfrc_header.h"	
DESCRIPTION	This function adds padding spaces to space out the header items above the test schedule field.	

FUNCTION	InputScheduleList	
PURPOSE	Opens the test schedule file and loads in the header items and parameter values to the test schedule field on the Comfrc interface	
SYNTAX	void InputScheduleList(GlobalPtr gp)	
	GlobalPtr gp	Global Comfrc pointer
INCLUDES	#include "comfrc_header.h"	
DESCRIPTION	This function opens the test schedule file and loads in the header items and parameter values to the test schedule field on the Comfrc interface. Sets the schedule position to the first item in the list. Sets the test schedule mode and apply tolerance checkbox on the Comfrc interface.	

FUNCTION	SetSchedulePosition	
PURPOSE	Increments the test schedule file list by the increment amount and highlights the current position	
SYNTAX	void SetSchedulePosition(Widget list_field, int nIncrement)	
	Widget list_field	Test Schedule field widget ID
	int nIncrement	Increment amount
INCLUDES	#include "comfrc_header.h"	
DESCRIPTION	This function increments the test schedule file list by the increment amount and highlights the current position	

Appendix C: MONFRC Software Functions

C.1. MONFRC Software Function Index

FUNCTION	DESCRIPTION	FILE
<u>act_gauge</u>	Draws the actuator graphic (if defined), and the actuator angle in degrees, according to the actuator initialisation file	acts.c
<u>act_overlay</u>	Draws the actuator description using a bitmap font in the overlay plane	acts.c
<u>CreateActDisplayLists</u>	Calls functions to create polygon display lists for any actuator graphics	acts.c
<u>DrawElevatorGraphic</u>	Draws a graphic of a PC9[1] tail with a moving elevator actuator	acts.c
<u>DrawRudderGraphic</u>	Draws a graphic of a PC9 tail with a moving rudder actuator	acts.c
<u>CreateAuxDisplayLists</u>	Calls functions to create polygon display lists for any graphics for the auxiliary module	aux.c
<u>rps_gauge</u>	Draws a rotating propeller graphic, rps value and units in the normal plane	aux.c
<u>rps_overlay</u>	Draws an RPS title using a bitmap font in the overlay plane	aux.c
<u>bar_gauge</u>	Draws the moving parts of a general graphical bar gauge with warning levels and a parameter value in the normal plane	bar.c
<u>bar_overlay</u>	Draws the bar gauge information required in the overlay plane including the gauge title, markings, scale and the warning level indicator	bar.c
<u>p_gauge</u>	Draws the animated parts of a mercury barometer gauge and the pressure value with units of kPa in the normal plane	fs.c
<u>p_overlay</u>	Draws the mercury barometer gauge information required in the overlay plane including the gauge title, markings and scale	fs.c

---

[1] Currently only the graphics for PC9 actuators are available. Actuator graphics for other models can be incorporated when required.

t_gauge	Draws the animated parts of a temperature gauge (mercury thermometer) and the temperature value with units of degrees C in the normal plane	fs.c
t_overlay	Draws the temperature gauge information required in the overlay plane including the gauge title, markings and scale	fs.c
v_gauge	Draws the animated parts of a velocity dial gauge with warning levels and the gauge value in m/s in the normal plane	fs.c
v_overlay	Draws the velocity dial gauge information required in the overlay plane including the gauge title, markings, scale and the warning level indicator	fs.c
angle_gauge	Draws the text value of another model angle (defined by the function arguments) with units of degrees in the normal plane (no graphic)	inc.c
angle_overlay	Draws the title (or description) of another model angle (defined by the function arguments) using a bitmap font in the overlay plane	inc.c
CreateIncDisplayLists	Calls functions to create polygon display lists for the graphics of the pitch, roll and yaw gauges - side, top and rear views of the model type	inc.c
pitch_gauge	Draws the animated parts of a model pitch dial gauge with warning levels and the gauge value with units of degrees in the normal plane	inc.c
pitch_overlay	Draws the pitch dial gauge information required in the overlay plane including the gauge title, markings, scale and the warning level indicator	inc.c
roll_gauge	Draws the animated parts of a model roll dial gauge with warning levels and the gauge value with units of degrees in the normal plane	inc.c
roll_overlay	Draws the roll dial gauge information required in the overlay plane including the gauge title, markings, scale and the warning level indicator	inc.c

yaw_gauge	Draws the animated parts of a model yaw dial gauge with warning levels and the gauge value with units of degrees in the normal plane	inc.c
yaw_overlay	Draws the yaw dial gauge information required in the overlay plane including the gauge title, markings, scale and the warning level indicator	inc.c
BitmapStrLen	Calculates the length in pixels of a string written with a GLUT bitmap font	monfrc.c
CalculateAlphaBeta	Calculates and sets alpha and beta values from theta, phi inputs	monfrc.c
CreatePolygonList	Creates a display list using the input array of (x,y) points to form a smooth shaded polygon using GLUT routines	monfrc.c
display	Draws on the screen all that is required in the "normal" plane	monfrc.c
drawOverlay	Draws on the screen all that is required in the "overlay" plane	monfrc.c
FlashGaugeAlarm	Flashes a red rectangle with vertices defined by the function arguments	monfrc.c
Init	Initialises GLUT lighting and colour conditions	monfrc.c
main	Main function for MONFRC	monfrc.c
output	Outputs message using the GLUT bitmap font specified, starting at location (x,y)	monfrc.c
Reshape	Updates the normal and overlay plane viewports and determines the coordinate system	monfrc.c
showMessage	Outputs message to the screen using the GLUT stroke font and the scale specified, starting at a location (x,y,z)	monfrc.c
StrokeStrLen	Calculates the length in pixels of a string written with a GLUT stroke font	monfrc.c
update	Acquires and converts the wind tunnel data to be displayed on the gauges and initiates a redraw of the display	monfrc.c
CalculateSGLoadsAndPercs	Calculates strain gauge balance loads and percentages of safe loads using data from the configuration file and strain gauge values	sgs.c

sg_gauge	Draws, in the normal plane, the animated parts of a bar gauge for a single strain gauge channel. The text values of the voltage and the balance load (lb/lbft) are also displayed.	sgs.c
sg_overlay1	Draws the information required in the overlay plane common to a group of strain gauge channels including the gauge markings, scale and labels	sgs.c
sg_overlay2	Draws the information required in the overlay plane for each individual strain gauge channel including the title and warning level indicator	sgs.c

C.2. MONFRC Software Function Summaries

FUNCTION	act_gauge	
PURPOSE	Draws the actuator graphic (if defined), and the actuator angle in degrees, according to the actuator initialisation file	
SYNTAX	void act_gauge(ActDataPtr adp);	
	ActDataPtr adp;	Pointer to a structure containing information about actuators
INCLUDES	#include "monfrc.h" #include "acts.h" #include "act_cntrl.h"	
DESCRIPTION	This function writes an actuator value in degrees in the normal plane for each valid actuator channel specified in the actuator initialisation file. If the model type is a PC9, the appropriate graphics for a rudder and elevator are also drawn.	

FUNCTION	act_overlay	
PURPOSE	Draws the actuator description using a bitmap font in the overlay plane	
SYNTAX	void act_overlay(ActDataPtr adp);	
	ActDataPtr adp;	Pointer to a structure containing information about actuators
INCLUDES	#include "monfrc.h" #include "acts.h" #include "act_cntrl.h"	
DESCRIPTION	This function obtains the actuator descriptions for all valid actuator channels from the actuator initialisation file and prints them in the overlay plane in a bitmap font.	

FUNCTION	CreateActDisplayLists
PURPOSE	Calls functions to create polygon display lists for any actuator graphics
SYNTAX	void CreateActDisplayLists();
INCLUDES	#include "monfrc.h" #include "acts.h"
DESCRIPTION	This function creates polygon display lists for any actuator graphics. Each graphic is defined by an array of points in this file and the display list created is named as a member of an ENUM list in monfrc.h.

FUNCTION	DrawElevatorGraphic	
PURPOSE	Draws a graphic of a PC9 tail with a moving elevator actuator	
SYNTAX	`void DrawElevatorGraphic(float ele_value);`	
	`float ele_value;`	Elevator angle in degrees
INCLUDES	`#include "monfrc.h"` `#include "acts.h"`	
DESCRIPTION	This function calls the display list ELEVATOR to draw a PC9 tail section with a moving elevator graphic in the normal plane.	

FUNCTION	DrawRudderGraphic	
PURPOSE	Draws a graphic of a PC9 tail with a moving rudder actuator	
SYNTAX	`void DrawRudderGraphic(float rud_value);`	
	`float rud_value;`	Rudder angle in degrees
INCLUDES	`#include "monfrc.h"` `#include "acts.h"`	
DESCRIPTION	This function calls the display list RUDDER to draw a PC9 tail section with a moving rudder graphic in the normal plane.	

FUNCTION	CreateAuxDisplayLists
PURPOSE	Calls functions to create polygon display lists for any graphics for the auxiliary module
SYNTAX	`void CreateAuxDisplayLists();`
INCLUDES	`#include "monfrc.h"` `#include "aux.h"`
DESCRIPTION	This function creates polygon display lists of any graphics for the auxiliary module. Each graphic is defined by an array of points in this file and the display list created is named as a member of an ENUM list in `monfrc.h`. At present, only a propeller graphic is available.

FUNCTION	rps_gauge	
PURPOSE	Draws a rotating propeller graphic, rps value and units in the normal plane	
SYNTAX	void rps_gauge(GaugeInfoPtr rps);	
	GaugeInfoPtr rps;	Pointer to a structure containing information about RPS gauge
INCLUDES	#include "monfrc.h" #include "aux.h"	
DESCRIPTION	This function calls the display list PROPELLER to draw a rotating propeller with an RPS value drawn in the normal plane.	

FUNCTION	rps_overlay	
PURPOSE	Draws an RPS title using a bitmap font in the overlay plane	
SYNTAX	void rps_overlay(GaugeInfoPtr rps);	
	GaugeInfoPtr rps;	Pointer to a structure containing information about RPS gauge
INCLUDES	#include "monfrc.h" #include "aux.h"	
DESCRIPTION	This function draws an RPS title using a bitmap font in the overlay plane.	

FUNCTION	bar_gauge	
PURPOSE	Draws the moving parts of a general graphical bar gauge with warning levels and a parameter value in the normal plane	
SYNTAX	void bar_gauge(float value, GaugeInfoPtr bar);	
	float value;	Value of bar gauge parameter
	GaugeInfoPtr bar;	Pointer to a structure containing information about the gauge
INCLUDES	#include "monfrc.h" #include "bar.h"	
DESCRIPTION	This function draws the moving graphical parts of a general bar gauge and the gauge value in text in the appropriate warning level colour in the normal plane.	

FUNCTION	bar_overlay	
PURPOSE	Draws the bar gauge information required in the overlay plane including the gauge title, markings, scale and the warning level indicator	
SYNTAX	`void rps_overlay(GaugeInfoPtr bar);`	
	`GaugeInfoPtr bar;`	Pointer to a structure containing information about the gauge
INCLUDES	`#include "monfrc.h"` `#include "bar.h"`	
DESCRIPTION	This function draws the bar gauge information required in the overlay plane including the gauge title, markings, scale and the warning level indicator.	

FUNCTION	p_gauge	
PURPOSE	Draws the animated parts of a mercury barometer gauge and the pressure value with units of kPa in the normal plane	
SYNTAX	`void p_gauge(GaugeInfoPtr p);`	
	`GaugeInfoPtr p;`	Pointer to a structure containing information about the pressure gauge
INCLUDES	`#include "monfrc.h"` `#include "fs.h"`	
DESCRIPTION	This function draws the animated parts of a mercury barometer gauge and the gauge value in text in the normal plane.	

FUNCTION	p_overlay
PURPOSE	Draws the mercury barometer gauge information required in the overlay plane including the gauge title, markings and scale
SYNTAX	void p_overlay(GaugeInfoPtr p);
	GaugeInfoPtr p; — Pointer to a structure containing information about the pressure gauge
INCLUDES	#include "monfrc.h" #include "fs.h"
DESCRIPTION	This function draws the mercury barometer gauge information required in the overlay plane including the gauge title, markings and scale. Presently, only a final alarm is associated with this gauge.

FUNCTION	t_gauge
PURPOSE	Draws the animated parts of a temperature gauge (mercury thermometer) and the temperature value with units of degrees C in the normal plane
SYNTAX	void t_gauge(GaugeInfoPtr t);
	GaugeInfoPtr t; — Pointer to a structure containing information about the temperature gauge
INCLUDES	#include "monfrc.h" #include "fs.h"
DESCRIPTION	This function draws the animated parts of a mercury thermometer gauge and the gauge value in text in the normal plane.

FUNCTION	t_overlay	
PURPOSE	Draws the temperature gauge information required in the overlay plane including the gauge title, markings and scale	
SYNTAX	void t_overlay(GaugeInfoPtr t);	
	GaugeInfoPtr t;	Pointer to a structure containing information about the temperature gauge
INCLUDES	#include "monfrc.h" #include "fs.h"	
DESCRIPTION	This function draws the thermometer gauge information required in the overlay plane including the gauge title, markings and scale. Presently, only a final alarm is associated with this gauge.	

FUNCTION	v_gauge	
PURPOSE	Draws the animated parts of a velocity dial gauge with warning levels and the gauge value in m/s in the normal plane	
SYNTAX	void v_gauge(GaugeInfoPtr v);	
	GaugeInfoPtr v;	Pointer to a structure containing information about the velocity dial gauge
INCLUDES	#include "monfrc.h" #include "fs.h"	
DESCRIPTION	This function draws the animated parts of a velocity dial gauge and displays the gauge value as text in the normal plane. The sector swept out by the gauge needle is of a solid colour associated with the warning level.	

FUNCTION	v_overlay
PURPOSE	Draws the velocity dial gauge information required in the overlay plane including the gauge title, markings, scale and the warning level indicator
SYNTAX	void v_overlay(GaugeInfoPtr v);
	GaugeInfoPtr v; — Pointer to a structure containing information about the velocity dial gauge
INCLUDES	#include "monfrc.h" #include "fs.h"
DESCRIPTION	This function draws the velocity dial gauge information required in the overlay plane including the gauge title, markings, scale and the warning level indicator.

FUNCTION	angle_gauge
PURPOSE	Draws the text value of another model angle (defined by the function arguments) with units of degrees in the normal plane (no graphic)
SYNTAX	void angle_gauge(float angle_value, GaugeInfoPtr angle);
	float angle_value; — Value of angle in degrees
	GaugeInfoPtr angle; — Pointer to a structure containing information about the angle gauge
INCLUDES	#include "monfrc.h" #include "inc.h"
DESCRIPTION	This function draws just the text value of another model angle (defined by the function arguments) in the normal plane. No graphic is associated with this gauge. No warning levels are associated with this gauge.

FUNCTION	angle_overlay	
PURPOSE	Draws the title (or description) of another model angle (defined by the function arguments) using a bitmap font in the overlay plane	
SYNTAX	`void angle_overlay(GaugeInfoPtr angle);`	
	`GaugeInfoPtr angle;`	Pointer to a structure containing information about the angle gauge
INCLUDES	`#include "monfrc.h"` `#include "inc.h"`	
DESCRIPTION	This function draws the title of another model angle (defined by the function arguments) in the overlay plane.	

FUNCTION	CreateIncDisplayLists	
PURPOSE	Calls functions to create polygon display lists for the graphics of the pitch, roll and yaw gauges – side, top and rear views of the model type	
SYNTAX	`void CreateIncDisplayLists();`	
INCLUDES	`#include "monfrc.h"` `#include "inc.h"`	
DESCRIPTION	This function creates polygon display lists of the graphics in the pitch, roll and yaw gauges – side, top and rear views of the model type. Each graphic is defined by an array of points in this file and the display list created is named as a member of an ENUM list in `monfrc.h`.	

FUNCTION	pitch_gauge	
PURPOSE	Draws the animated parts of a model pitch dial gauge with warning levels and the gauge value with units of degrees in the normal plane	
SYNTAX	`void pitch_gauge(GaugeInfoPtr pitch);`	
	`GaugeInfoPtr pitch;`	Pointer to a structure containing information about the pitch dial gauge
INCLUDES	`#include "monfrc.h"` `#include "inc.h"`	
DESCRIPTION	This function draws the animated parts of a pitch dial gauge and the gauge value in text in the normal plane. A side view graphic of the wind tunnel model is displayed with the appropriate pitch angle at the centre of the gauge. The colours of this graphic and the text value are associated with the warning level.	

FUNCTION	pitch_overlay
PURPOSE	Draws the pitch dial gauge information required in the overlay plane including the gauge title, markings, scale and the warning level indicator
SYNTAX	`void pitch_overlay(GaugeInfoPtr pitch);`
	`GaugeInfoPtr pitch;` Pointer to a structure containing information about the pitch dial gauge
INCLUDES	`#include "monfrc.h"` `#include "inc.h"`
DESCRIPTION	This function draws the pitch dial gauge information required in the overlay plane including the gauge title, markings, scale and the warning level indicator.

FUNCTION	roll_gauge
PURPOSE	Draws the animated parts of a model roll dial gauge with warning levels and the gauge value with units of degrees in the normal plane
SYNTAX	`void roll_gauge(GaugeInfoPtr roll);`
	`GaugeInfoPtr roll;` Pointer to a structure containing information about the roll dial gauge
INCLUDES	`#include "monfrc.h"` `#include "inc.h"`
DESCRIPTION	This function draws the animated parts of a roll dial gauge and the gauge value in text in the normal plane. A rear view graphic of the wind tunnel model is displayed with the appropriate roll angle at the centre of the gauge. The colours of this graphic and the text value are associated with the warning level.

FUNCTION	roll_overlay	
PURPOSE	Draws the roll dial gauge information required in the overlay plane including the gauge title, markings, scale and the warning level indicator	
SYNTAX	void roll_overlay(GaugeInfoPtr roll);	
	GaugeInfoPtr roll;	Pointer to a structure containing information about the roll dial gauge
INCLUDES	#include "monfrc.h" #include "inc.h"	
DESCRIPTION	This function draws the roll dial gauge information required in the overlay plane including the gauge title, markings, scale and the warning level indicator.	

FUNCTION	yaw_gauge	
PURPOSE	Draws the animated parts of a model yaw dial gauge with warning levels and the gauge value with units of degrees in the normal plane	
SYNTAX	void yaw_gauge(GaugeInfoPtr yaw);	
	GaugeInfoPtr yaw;	Pointer to a structure containing information about the yaw dial gauge
INCLUDES	#include "monfrc.h" #include "inc.h"	
DESCRIPTION	This function draws the animated parts of a yaw dial gauge and the gauge value in text in the normal plane. A top view graphic of the wind tunnel model is displayed with the appropriate yaw angle at the centre of the gauge. The colours of this graphic and the text value are associated with the warning level.	

FUNCTION	yaw_overlay	
PURPOSE	Draws the yaw dial gauge information required in the overlay plane including the gauge title, markings, scale and the warning level indicator	
SYNTAX	`void yaw_overlay(GaugeInfoPtr yaw);`	
	`GaugeInfoPtr yaw;`	Pointer to a structure containing information about the yaw dial gauge
INCLUDES	`#include "monfrc.h"` `#include "inc.h"`	
DESCRIPTION	This function draws the yaw dial gauge information required in the overlay plane including the gauge title, markings, scale and the warning level indicator.	

FUNCTION	BitmapStrLen	
PURPOSE	Calculates the length in pixels of a string written with a GLUT bitmap font	
SYNTAX	`int BitmapStrLen(char *s, void *bitmapfont);`	
	`char *s;`	String whose length is to be measured
	`void *bitmapfont;`	GLUT bitmap font structure
INCLUDES	`#include "monfrc.h"`	
DESCRIPTION	This function calculates the length in pixels of a string written with a GLUT bitmap font. This allows the correct positioning of such a string on the screen. The function returns the length in pixels of the string as an `int`.	

FUNCTION	CalculateAlphaBeta	
PURPOSE	Calculates and sets alpha and beta values from theta, phi inputs	
SYNTAX	`void CalculateAlphaBeta(float theta, float phi, float *alpha, float *beta);`	
	`float theta;`	Theta in radians
	`float phi;`	Phi in radians
	`float *alpha;`	Returned value of Alpha in radians
	`float *beta;`	Returned value of Beta in radians
INCLUDES	`#include "monfrc.h"`	
DESCRIPTION	This function simply calculates the aerodynamic angles Alpha and Beta from Theta and Phi assuming Psi = 0 with the equations: $\alpha = \sin^{-1}(\sin\theta\cos\phi)$ $\beta = \sin^{-1}(\sin\theta\sin\phi)$ where α = Alpha β = Beta θ = Theta ϕ = Phi	

FUNCTION	CreatePolygonList	
PURPOSE	Creates a display list using the input array of (x,y) points to form a smooth shaded polygon using GLUT routines	
SYNTAX	`void CreatePolygonList(float data[][2], unsigned int dataSize, int listname);`	
	`float data[][2];`	Array of (x,y) points to form a polygon
	`unsigned int *dataSize;`	`sizeof` data
	`int listname;`	Integer to identify name of the list being created
INCLUDES	`#include "monfrc.h"`	
DESCRIPTION	This function creates a display list of points defining a polygon for quick drawing of a graphic to the screen. Each graphic is identified by the `listname` integer, which is in an enum list called `displayLists` in "monfrc.h".	

FUNCTION	display
PURPOSE	Draws on the screen all that is required in the "normal" plane
SYNTAX	`void display(void);`
INCLUDES	`#include "monfrc.h"`
DESCRIPTION	This function is the argument for a standard GLUT routine called `glutDisplayFunc`, which calls many routines that draw all that is required in the "normal" plane. The graphics on this plane are redrawn for each screen update. Therefore, any portions of the graphical gauges or text that move or change are drawn in this plane.

FUNCTION	drawOverlay
PURPOSE	Draws on the screen all that is required in the "overlay" plane
SYNTAX	`void drawOverlay(void);`
INCLUDES	`#include "monfrc.h"`
DESCRIPTION	This function is the argument for a standard GLUT routine called `glutOverlayDisplayFunc`, which involves the drawing of everything on the overlay plane. The graphics on this plane are drawn once when the program is started or when the screen is hidden or obscured and a redraw is required. Therefore, any portions of the graphical gauges that do not change are drawn in this plane including titles and gauge markings.

FUNCTION	FlashGaugeAlarm	
PURPOSE	Flashes a red rectangle with vertices defined by the function arguments	
SYNTAX	`void FlashGaugeAlarm(float x1, float y1,` `float x2, float y2);`	
	`float x1;`	X coordinate of the first corner of the rectangle
	`float y1;`	Y coordinate of the first corner of the rectangle
	`float x2;`	X coordinate of the second corner of the rectangle, diagonally opposite to the first corner
	`float y2;`	Y coordinate of the second corner of the rectangle, diagonally opposite to the first corner
INCLUDES	`#include "monfrc.h"`	
DESCRIPTION	This is a simple function to flash a red rectangle on the screen as the background to a gauge for an alarm. It requires a global variable, `static int c`, which is declared outside this function at the top of the `monfrc.c` file. This variable continuously cycles between `-flash` and `+flash` (a #define in `"monfrc.h"`) with every update of the screen performed in the function `void update(void)`. If the variable c is positive only then is the red rectangle drawn. This produces the flashing effect.	

FUNCTION	Init
PURPOSE	Initialises GLUT lighting and colour conditions
SYNTAX	`void Init(void);`
INCLUDES	`#include "monfrc.h"`
DESCRIPTION	This function calls routines from the GLUT/OpenGL libraries to initialise the lighting, colour conditions and shading models for the screen.

FUNCTION	main
PURPOSE	Main function for MONFRC
SYNTAX	`int main(int argc, char *argv[]);`
INCLUDES	`#include "monfrc.h"`
DESCRIPTION	This function performs several initialisation tasks: 1. An attempt to open a connection to the shared memory database is made. If the connection fails an error message appears and the application is terminated. 2. If the option `-c config_file.dat` is specified during startup, this configuration file is used instead of the default `conf_new.dat`. 3. The contents of the configuration file are read. 4. The contents of the MONFRC initialisation file, `monfrc.ini`, are read. 5. LSWT Modules are set to streaming mode ON 6. Freestream and Inclinometer Modules are initialised, using `initFs` and `initInc`. 7. Actuator initialisation is performed if required. 8. Model attitude and orientation gauge data is set according to the configuration file if necessary. 9. The display window size and other characteristics are initialised. 10. Display lists are created for the graphics of the actuator, model attitude and orientation and auxiliary gauges. 11. The callbacks, `Reshape`, `display` and `update` are established. 12. An attempt to establish an overlay plane is made. If this is successful, colour mapping for the overlay plane is performed and a callback for the overlay display, `drawOverlay`, is established. 13. The main loop is entered.

FUNCTION	output	
PURPOSE	Outputs message using the GLUT bitmap font specified, starting at location (x,y)	
SYNTAX	`void output(int x, int y, char *string, void *bitmapfont);`	
	`int x;`	Starting X position
	`int y;`	Starting Y position
	`char *string;`	String to be displayed as output
	`void *bitmapfont;`	GLUT bitmap font structure
INCLUDES	`#include "monfrc.h"`	
DESCRIPTION	This function outputs to the screen starting from a position (x,y) a string in a GLUT bitmap font	

FUNCTION	Reshape	
PURPOSE	Updates the normal and overlay plane viewports and determines the coordinate system	
SYNTAX	`void Reshape(int width, int height);`	
	`int width;`	Width of viewport
	`int height;`	Height of viewport
INCLUDES	`#include "monfrc.h"`	
DESCRIPTION	This function is the argument for a standard GLUT routine called `glutReshapeFunc`, and sets the viewport of the normal and overlay planes to be the size of the entire window. It also sets a projection matrix such that drawing is performed in an orthographic system with window-relative coordinates and an origin positioned in the lower left corner.	

FUNCTION	showMessage	
PURPOSE	Outputs message to the screen using the GLUT stroke font and the scale specified, starting at a location (x,y,z)	
SYNTAX	`void showMessage(float x, float y, float z,` `float scale, char *message, void *strokefont);`	
	`float x;`	Starting Y position
	`float y;`	Starting Y position
	`float x;`	Starting Z position
	`float scale;`	Scale factor of the stroke font
	`char *message;`	String to be shown on the screen
	`void *strokefont`	GLUT stroke font structure.
INCLUDES	`#include "monfrc.h"`	
DESCRIPTION	This function displays on the screen starting from a position (x,y,z) a string in a GLUT stroke font with the scale specified	

FUNCTION	StrokeStrLen	
PURPOSE	Calculates the length in pixels of a string written with a GLUT stroke font	
SYNTAX	`int StrokeStrLen(char *s, void *strokefont,` `float scale);`	
	`char *s;`	String whose length is to be measured
	`void *strokefont;`	GLUT stroke font structure
	`float scale;`	Scale factor of the font
INCLUDES	`#include "monfrc.h"`	
DESCRIPTION	This function calculates the length in pixels of a string written with a GLUT stroke font. This allows the correct positioning of such a string on the screen. The function returns the length in pixels of the string as an `int`.	

FUNCTION	update
PURPOSE	Acquires and converts the wind tunnel data to be displayed on the gauges and initiates a redraw of the display
SYNTAX	`void update(void);`

INCLUDES	`#include "monfrc.h"`
	`#include "DataAcqFuncs.h"`
	`#include "act_cntrl.h"`

DESCRIPTION	This function is the argument for a standard GLUT routine called `glutIdleFunc`, and is continuously called during any idle CPU time. During this function call, a counter is incremented between `-flash` and `+flash` for the purpose of gauge alarms as described in the function `FlashGaugeAlarm`. It then acquires wind tunnel information from the shared memory database and converts it to engineering units. Appropriate model attitude and orientation angles are calculated as are strain gauge loads and percentages. By putting a `glutPostRedisplay` command in this function, the `display` function is recalled, which performs a screen update and redraws all that is on the normal plane. This creates the animated effect of the gauges.

FUNCTION	CalculateSGLoadsAndPercs	
PURPOSE	Calculates strain gauge balance loads and percentages of safe loads using data from the configuration file and strain gauge values	
SYNTAX	`void CalculateSGLoadsAndPercs(ConfigDataPtr cdp, ZeroDataPtr zdp, float sg_value[6], float sg_load[6], float sg_perc[6]);`	
	`ConfigDataPtr cdp;`	Pointer to a structure containing configuration data for the wind tunnel test
	`ZeroDataPtr zdp;`	Pointer to a structure containing first and final zeros information
	`float sg_value[6];`	Strain Gauge voltages in volts
	`float sg_load[6];`	Strain Gauge balance loads in lb/lbft
	`float sg_perc[6];`	Loads as a percentage of the balance safe loads
INCLUDES	`#include "monfrc.h"` `#include "sgs.h"`	
DESCRIPTION	This function calculates the strain gauge balance loads and percentages of safe loads using data from the configuration file and strain gauge values. The function `balred` is called which calculates the balance loads using the calibration matrices in the configuration file. The safeloads in the configuration file are used to calculate the percentages.	

FUNCTION	sg_gauge	
PURPOSE	Draws, in the normal plane, the animated parts of a bar gauge for a single strain gauge channel. The text values of the voltage and the balance load (lb/lbft) are also displayed.	
SYNTAX	void sg_gauge(int ch, float sg_value, float sg_load, float sg_perc, GaugeInfoPtr sg);	
	int ch;	Channel number
	float sg_value;	Strain Gauge voltage (Volts)
	float sg_load;	Strain Gauge balance load (lb/lbft)
	float sg_perc;	Load as percentage of balance safe loads
	GaugeInfoPtr sg;	Pointer to a structure containing information about the strain gauges
INCLUDES	#include "monfrc.h" #include "sgs.h"	
DESCRIPTION	This function draws, in the normal plane, the animated parts of a bar gauge for a single strain gauge channel. The gauge value is the percentage of the balance safe load. The text values of the voltage and the balance load (lb/lbft) are also displayed. The colour of the bar and text values are associated with the warning levels. If a strain gauge amplifier overloads (10 Volts) an overload (O/L) message appears in red under the gauge.	

FUNCTION	sg_overlay1	
PURPOSE	Draws the information required in the overlay plane common to a group of strain gauge channels including the gauge markings, scale and labels	
SYNTAX	void sg_overlay1(GaugeInfoPtr sg);	
	GaugeInfoPtr sg;	Pointer to a structure containing information about the strain gauges
INCLUDES	#include "monfrc.h" #include "sgs.h"	
DESCRIPTION	This function draws the information required in the overlay plane common to a group of strain gauge channels including the gauge markings, scale and the labels on the first gauge only.	

FUNCTION	sg_overlay2
PURPOSE	Draws the information required in the overlay plane for each individual strain gauge channel including the title and warning level indicator
SYNTAX	`void sg_overlay2(char *title, GaugeInfoPtr sg);`
	`char *title;` · Individual title for the strain gauge channel
	`GaugeInfoPtr sg;` · Pointer to a structure containing information about the strain gauges
INCLUDES	`#include "monfrc.h"` `#include "sgs.h"`
DESCRIPTION	This function draws the information required in the overlay plane for each individual strain gauge channel including the title and warning level indicator.

Appendix D: CALIB Software Functions

D.1. CALIB Software Function Index

FUNCTION	DESCRIPTION	FILE
buttonCB	Callback function for the 'Select Model' button	callback.c
button2CB	Callback function for the 'Select data input file' button	callback.c
button3CB	Callback function for the 'Select data output file' button	callback.c
button_goCB	Callback function for the 'button_go' button (Execute Button) in FORM6	callback.c
button_quitCB	Callback function for the 'button_quit' button (Quit Button) in FORM6	callback.c
changeCB1_1	Callback function for the toggle button (Multivariable Regression) in FORM2	callback.c
changeCB1_2	Callback function for the toggle button (Ramaswamy's Least Squares Method) in FORM2	callback.c
changeCB2_1	Callback function for the toggle button (No Optimisation) in FORM4	callback.c
changeCB2_2	Callback function for the toggle button (1se Optimisation) in FORM4	callback.c
changeCB2_3	Callback function for the toggle button (2se Optimisation) in FORM4	callback.c
changeCB2_4	Callback function for the toggle button (3se Optimisation) in FORM4	callback.c
changeCB3_1	Callback function for the toggle button (Activate reverse optimisation) in FORM5	callback.c
changeCB3_2	Callback function for the toggle button (De-activate reverse optimisation) in FORM5	callback.c
dialogCB	Popup dialog list for calibration model selection	callback.c
fdialog1CB	Callback function for the 'fdialog1' file selection dialog box in FORM1	callback.c
fdialog2CB	Callback function for the 'fdialog2' file selection dialog box in FORM1	callback.c
text1_1CB	Callback function for the 'text1_1' text box ('Zero' filter range [%]) in FORM1	callback.c

text2CB	Callback function for the 'text2' text box in FORM3	callback.c
text3CB	Callback function for the 'text3' text box in FORM3	callback.c
toggle1_1	Callback function for the 'toggle1_1' toggle button (Activate 'zero' data points filter) in FORM1	callback.c
toggle7_1	Callback function for the toggle button 'toggle7_1' (Activate Chauvenet's Criterion) in FORM7	callback.c
toggleCB	Callback function for the toggle button (Tolerances in R and H) in FORM3	callback.c
chauvenet	Performs Chauvenet's Criterion on a set of data	chauvenet.c
coeff	Generate an output file which contains the result obtained by CALIB	coeff.c
convert5to6	Convert a 5 components coefficient matrix to a 6 components coefficient matrix	convert5to6.c
dataout	Output a text file which contains the calibration data set which will be used by CALIB	dataout.c
filter	Performs zero filtering procedure to eliminate close to zero values from the original calibration data set.	filter.c
gaussj	Performs Gauss-Jordan elimination with full pivoting	gaussj.c
leastsquare1	Calculate the balance calibration coefficient matrix using the multivariable regression least squares method.	leastsquare1.c
leastsquare2_1	Calculate the balance calibration coefficient matrix for the calibration model: [H]=[C][R-H] using the Ramaswamy least squares method.	leastsquare2_1.c
leastsquare2_2	Calculate the balance calibration coefficient matrix for the calibration model: [R]=[C][H] using the Ramaswamy least squares method.	leastsquare2_2.c
leastsquare2_3	Calculate the balance calibration coefficient matrix for the calibration model: [H]=[C][R] using the Ramaswamy least squares method.	leastsquare2_3.c
optimise	Eliminate any outliers defined by the user from the original calibration data set.	optimise.c
rev_calib1	Performs reverse calibration on calibration model: [H]=[C][R-H].	rev_calib1.c
rev_calib2	Performs reverse calibration on calibration model: [R]=[C][H].	rev_calib2.c

rev_calib3	Performs reverse calibration on calibration model: [H]=[C][R].	rev_calib3.c
rfile	Reads the data input file and the balance load range file	rfile.c
rfile2	Reads the balance design load range from the input data file (des_load.dat)	rfile2.c
RR_model1	Generates RR2 matrix for second and third order calibration equations for the calibration model: [H]=[C][R-H]	RR_model1.c
RR_model2	Generates RR2 matrix for second and third order calibration equations for the calibration model: [R]=[C][H]	RR_model2.c
RR_model3	Generates RR2 matrix for second and third order calibration equations for the calibration model: [H]=[C][R]	RR_model3.c
statistic	Calculates the standard errors and the coefficient of multiple correlation for each component of the strain gauge balance described by the balance calibration equation	statistic.c

D.2. CALIB Software Function Summaries

FUNCTION	buttonCB	
PURPOSE	Callback function for the 'Select Model' button	
SYNTAX	void buttonCB(w, client_data, call_data)	
	Widget w	Parent widget ID
	CALIB *client_data;	A structure which contains the variables used to define the dialog list
	XmPushButtonCallbackStruct *call_data	Specialised structure with information useful to the function
INCLUDES	#include <Xm/Xm.h> #include <Xm/Form.h> #include <Xm/Label.h> #include <Xm/PushB.h> #include <Xm/SelectionB.h> #include <Xm/List.h> #include <Xm/RowColumn.h> #include <Xm/ToggleB.h> #include <Xm/Frame.h> #include <Xm/Text.h> #include <Xm/Separator.h> #include <Xm/FileSB.h> #include <Xm/MessageB.h> #include "structure.h" #include "variables.h" #include "calib_function.h"	
DESCRIPTION	Provides callback functionality to the 'Select Model' button in the GUI interface	

FUNCTION	button2CB	
PURPOSE	Callback function for the 'Select data input file' button	
SYNTAX	void button2CB(w, client_data, call_data)	
	Widget w	Parent widget ID
	CALIB *client_data;	A structure which contains the variables used to define the dialog list
	XmPushButtonCallbackStruct *call_data	Specialised structure with information useful to the function
INCLUDES	#include <Xm/Xm.h> #include <Xm/Form.h> #include <Xm/Label.h> #include <Xm/PushB.h> #include <Xm/SelectionB.h> #include <Xm/List.h> #include <Xm/RowColumn.h> #include <Xm/ToggleB.h> #include <Xm/Frame.h> #include <Xm/Text.h> #include <Xm/Separator.h> #include <Xm/FileSB.h> #include <Xm/MessageB.h> #include "structure.h" #include "variables.h" #include "calib_function.h"	
DESCRIPTION	Provides callback functionality to the 'Select data input file' button in the GUI interface	

FUNCTION	button3CB	
PURPOSE	Callback function for the 'Select data output file' button	
SYNTAX	void button3CB(w, client_data, call_data)	
	Widget w	Parent widget ID
	CALIB *client_data;	A structure which contains the variables used to define the dialog list
	XmPushButtonCallbackStruct *call_data	Specialised structure with information useful to the function
INCLUDES	#include <Xm/Xm.h> #include <Xm/Form.h> #include <Xm/Label.h> #include <Xm/PushB.h> #include <Xm/SelectionB.h> #include <Xm/List.h> #include <Xm/RowColumn.h> #include <Xm/ToggleB.h> #include <Xm/Frame.h> #include <Xm/Text.h> #include <Xm/Separator.h> #include <Xm/FileSB.h> #include <Xm/MessageB.h> #include "structure.h" #include "variables.h" #include "calib_function.h"	
DESCRIPTION	Provides callback functionality to the 'Select data output file' button in the GUI interface	

FUNCTION	button_goCB	
PURPOSE	Callback function for the 'button_go' button (Execute Button) in FORM6	
SYNTAX	void button_goCB(w, client_data, call_data)	
	Widget w	Parent widget ID
	CALIB *client_data;	A structure which contains the variables used to define the dialog list
	XmPushButtonCallbackStruct *call_data	Specialised structure with information useful to the function
INCLUDES	#include <Xm/Xm.h #include <Xm/Form.h #include <Xm/Label.h #include <Xm/PushB.h #include <Xm/SelectionB.h #include <Xm/List.h #include <Xm/RowColumn.h #include <Xm/ToggleB.h #include <Xm/Frame.h #include <Xm/Text.h #include <Xm/Separator.h #include <Xm/FileSB.h #include <Xm/MessageB.h #include "structure.h" #include "variables.h" #include "calib_function.h"	
DESCRIPTION	Provides callback functionality to the 'button_go' button (Execute Button) in the GUI interface. This function executes various functions in the program based on the user's selection in the GUI interface.	

FUNCTION	button_quitCB	
PURPOSE	Callback function for the 'button_quit' button (Quit Button) in FORM6	
SYNTAX	void button_quitCB(w, client_data, call_data)	
	Widget w	Parent widget ID
	CALIB *client_data;	A structure which contains the variables used to define the dialog list
	XmPushButtonCallbackStruct *call_data	Specialised structure with information useful to the function
INCLUDES	#include <Xm/Xm.h> #include <Xm/Form.h> #include <Xm/Label.h> #include <Xm/PushB.h> #include <Xm/SelectionB.h> #include <Xm/List.h> #include <Xm/RowColumn.h> #include <Xm/ToggleB.h> #include <Xm/Frame.h> #include <Xm/Text.h> #include <Xm/Separator.h> #include <Xm/FileSB.h> #include <Xm/MessageB.h> #include "structure.h" #include "variables.h" #include "calib_function.h"	
DESCRIPTION	Provides callback functionality to the 'button_quit' button (Quit Button) in the GUI interface. This function quit the program.	

FUNCTION	changeCB1_1	
PURPOSE	Callback function for the toggle button (Multivariable Regression) in FORM2	
SYNTAX	void changeCB1_1(w, client_data, call_data)	
	Widget w	Parent widget ID
	CALIB *client_data;	A structure which contains the variables used to define the dialog list
	XmToggleButtonCallbackStruct *call_data	Specialised structure with information useful to the function
INCLUDES	#include <Xm/Xm.h> #include <Xm/Form.h> #include <Xm/Label.h> #include <Xm/PushB.h> #include <Xm/SelectionB.h> #include <Xm/List.h> #include <Xm/RowColumn.h> #include <Xm/ToggleB.h> #include <Xm/Frame.h> #include <Xm/Text.h> #include <Xm/Separator.h> #include <Xm/FileSB.h> #include <Xm/MessageB.h> #include "structure.h" #include "variables.h" #include "calib_function.h"	
DESCRIPTION	Provides callback functionality to the toggle button (Multivariable Regression) in the GUI interface	

FUNCTION	changeCB1_2	
PURPOSE	Callback function for the toggle button (Ramaswamy's Least Squares Method) in FORM2	
SYNTAX	void changeCB1_2(w, client_data, call_data)	
	Widget w	Parent widget ID
	CALIB *client_data;	A structure which contains the variables used to define the dialog list
	XmToggleButtonCallbackStruct *call_data	Specialised structure with information useful to the function
INCLUDES	#include <Xm/Xm.h> #include <Xm/Form.h> #include <Xm/Label.h> #include <Xm/PushB.h> #include <Xm/SelectionB.h> #include <Xm/List.h> #include <Xm/RowColumn.h> #include <Xm/ToggleB.h> #include <Xm/Frame.h> #include <Xm/Text.h> #include <Xm/Separator.h> #include <Xm/FileSB.h> #include <Xm/MessageB.h> #include "structure.h" #include "variables.h" #include "calib_function.h"	
DESCRIPTION	Provides callback functionality to the toggle button (Ramaswamy's Least Squares Method) in the GUI interface	

FUNCTION	changeCB2_1	
PURPOSE	Callback function for the toggle button (No Optimisation) in FORM4	
SYNTAX	void changeCB2_1(w, client_data, call_data)	
	Widget w	Parent widget ID
	CALIB *client_data;	A structure which contains the variables used to define the dialog list
	XmToggleButtonCallbackStruct *call_data	Specialised structure with information useful to the function
INCLUDES	#include <Xm/Xm.h> #include <Xm/Form.h> #include <Xm/Label.h> #include <Xm/PushB.h> #include <Xm/SelectionB.h> #include <Xm/List.h> #include <Xm/RowColumn.h> #include <Xm/ToggleB.h> #include <Xm/Frame.h> #include <Xm/Text.h> #include <Xm/Separator.h> #include <Xm/FileSB.h> #include <Xm/MessageB.h> #include "structure.h" #include "variables.h" #include "calib_function.h"	
DESCRIPTION	Provides callback functionality to the toggle button (No Optimisation) in the GUI interface	

FUNCTION	changeCB2_2	
PURPOSE	Callback function for the toggle button (1se Optimisation) in FORM4	
SYNTAX	void changeCB2_2(w, client_data, call_data)	
	Widget w	Parent widget ID
	CALIB *client_data;	A structure which contains the variables used to define the dialog list
	XmToggleButtonCallbackStruct *call_data	Specialised structure with information useful to the function
INCLUDES	#include <Xm/Xm.h> #include <Xm/Form.h> #include <Xm/Label.h> #include <Xm/PushB.h> #include <Xm/SelectionB.h> #include <Xm/List.h> #include <Xm/RowColumn.h> #include <Xm/ToggleB.h> #include <Xm/Frame.h> #include <Xm/Text.h> #include <Xm/Separator.h> #include <Xm/FileSB.h> #include <Xm/MessageB.h> #include "structure.h" #include "variables.h" #include "calib_function.h"	
DESCRIPTION	Provides callback functionality to the toggle button (1se Optimisation) in the GUI interface	

FUNCTION	changeCB2_3	
PURPOSE	Callback function for the toggle button (2se Optimisation) in FORM4	
SYNTAX	`void changeCB2_3(w, client_data, call_data)`	
	`Widget w`	Parent widget ID
	`CALIB *client_data;`	A structure which contains the variables used to define the dialog list
	`XmToggleButtonCallbackStruct *call_data`	Specialised structure with information useful to the function

INCLUDES	`#include <Xm/Xm.h>`
	`#include <Xm/Form.h>`
	`#include <Xm/Label.h>`
	`#include <Xm/PushB.h>`
	`#include <Xm/SelectionB.h>`
	`#include <Xm/List.h>`
	`#include <Xm/RowColumn.h>`
	`#include <Xm/ToggleB.h>`
	`#include <Xm/Frame.h>`
	`#include <Xm/Text.h>`
	`#include <Xm/Separator.h>`
	`#include <Xm/FileSB.h>`
	`#include <Xm/MessageB.h>`
	`#include "structure.h"`
	`#include "variables.h"`
	`#include "calib_function.h"`	

DESCRIPTION	Provides callback functionality to the toggle button (2se Optimisation) in the GUI interface	

FUNCTION	changeCB2_4	
PURPOSE	Callback function for the toggle button (3se Optimisation) in FORM4	
SYNTAX	void changeCB2_4(w, client_data, call_data)	
	Widget w	Parent widget ID
	CALIB *client_data;	A structure which contains the variables used to define the dialog list
	XmToggleButtonCallbackStruct *call_data	Specialised structure with information useful to the function
INCLUDES	#include <Xm/Xm.h> #include <Xm/Form.h> #include <Xm/Label.h> #include <Xm/PushB.h> #include <Xm/SelectionB.h> #include <Xm/List.h> #include <Xm/RowColumn.h> #include <Xm/ToggleB.h> #include <Xm/Frame.h> #include <Xm/Text.h> #include <Xm/Separator.h> #include <Xm/FileSB.h> #include <Xm/MessageB.h> #include "structure.h" #include "variables.h" #include "calib_function.h"	
DESCRIPTION	Provides callback functionality to the toggle button (3se Optimisation) in the GUI interface	

140

FUNCTION	changeCB3_1	
PURPOSE	Callback function for the toggle button (Activate reverse optimisation) in FORM5	
SYNTAX	void changeCB3_1(w, client_data, call_data)	
	Widget w	Parent widget ID
	CALIB *client_data;	A structure which contains the variables used to define the dialog list
	XmToggleButtonCallbackStruct *call_data	Specialised structure with information useful to the function
INCLUDES	#include <Xm/Xm.h #include <Xm/Form.h #include <Xm/Label.h #include <Xm/PushB.h #include <Xm/SelectionB.h #include <Xm/List.h #include <Xm/RowColumn.h #include <Xm/ToggleB.h #include <Xm/Frame.h #include <Xm/Text.h #include <Xm/Separator.h #include <Xm/FileSB.h #include <Xm/MessageB.h #include "structure.h" #include "variables.h" #include "calib_function.h"	
DESCRIPTION	Provides callback functionality to the toggle button (Activate reverse optimisation) in the GUI interface	

FUNCTION	changeCB3_2	
PURPOSE	Callback function for the toggle button (De-activate reverse optimisation) in FORM5	
SYNTAX	void changeCB3_2(w, client_data, call_data)	
	Widget w	Parent widget ID
	CALIB *client_data;	A structure which contains the variables used to define the dialog list
	XmToggleButtonCallbackStruct *call_data	Specialised structure with information useful to the function
INCLUDES	#include <Xm/Xm.h> #include <Xm/Form.h> #include <Xm/Label.h> #include <Xm/PushB.h> #include <Xm/SelectionB.h> #include <Xm/List.h> #include <Xm/RowColumn.h> #include <Xm/ToggleB.h> #include <Xm/Frame.h> #include <Xm/Text.h> #include <Xm/Separator.h> #include <Xm/FileSB.h> #include <Xm/MessageB.h> #include "structure.h" #include "variables.h" #include "calib_function.h"	
DESCRIPTION	Provides callback functionality to the toggle button (De-activate reverse optimisation) in the GUI interface	

FUNCTION	dialogCB	
PURPOSE	Popup dialog list for calibration model selection	
SYNTAX	void dialogCB(w, client_data, call_data)	
	Widget w	Parent widget ID
	CALIB *client_data;	A structure which contains the variables used to define the dialog list
	XmSelectionBoxCallbackStruct *call_data	Specialised structure with information useful to the function

INCLUDES	`#include <Xm/Xm.h>`
	`#include <Xm/Form.h>`
	`#include <Xm/Label.h>`
	`#include <Xm/PushB.h>`
	`#include <Xm/SelectionB.h>`
	`#include <Xm/List.h>`
	`#include <Xm/RowColumn.h>`
	`#include <Xm/ToggleB.h>`
	`#include <Xm/Frame.h>`
	`#include <Xm/Text.h>`
	`#include <Xm/Separator.h>`
	`#include <Xm/FileSB.h>`
	`#include <Xm/MessageB.h>`
	`#include "structure.h"`
	`#include "variables.h"`
	`#include "calib_function.h"`	

DESCRIPTION	Construct a popup list for calibration model selection. The popup list contains all the calibration equations available in the program for analysis.	

FUNCTION	fdialog1CB	
PURPOSE	Callback function for the 'fdialog1' file selection dialog box in FORM1	
SYNTAX	void fdialog1CB(w, client_data, call_data)	
	Widget w	Parent widget ID
	CALIB *client_data;	A structure which contains the variables used to define the dialog list
	XmSelectionBoxCallbackStruct *call_data	Specialised structure with information useful to the function
INCLUDES	#include <Xm/Xm.h> #include <Xm/Form.h> #include <Xm/Label.h> #include <Xm/PushB.h> #include <Xm/SelectionB.h> #include <Xm/List.h> #include <Xm/RowColumn.h> #include <Xm/ToggleB.h> #include <Xm/Frame.h> #include <Xm/Text.h> #include <Xm/Separator.h> #include <Xm/FileSB.h> #include <Xm/MessageB.h> #include "structure.h" #include "variables.h" #include "calib_function.h"	
DESCRIPTION	Provides callback functionality to the 'fdialog1' file selection dialog box in the GUI interface	

FUNCTION	fdialog2CB	
PURPOSE	Callback function for the 'fdialog2' file selection dialog box in FORM1	
SYNTAX	`void fdialog2CB(w, client_data, call_data)`	
	`Widget w`	Parent widget ID
	`CALIB *client_data;`	A structure which contains the variables used to define the dialog list
	`XmSelectionBoxCallbackStruct *call_data`	Specialised structure with information useful to the function

INCLUDES	`#include <Xm/Xm.h`
	`#include <Xm/Form.h`
	`#include <Xm/Label.h`
	`#include <Xm/PushB.h`
	`#include <Xm/SelectionB.h`
	`#include <Xm/List.h`
	`#include <Xm/RowColumn.h`
	`#include <Xm/ToggleB.h`
	`#include <Xm/Frame.h`
	`#include <Xm/Text.h`
	`#include <Xm/Separator.h`
	`#include <Xm/FileSB.h`
	`#include <Xm/MessageB.h`
	`#include "structure.h"`
	`#include "variables.h"`
	`#include "calib_function.h"`	

DESCRIPTION	Provides callback functionality to the 'fdialog2' file selection dialog box in the GUI interface	

FUNCTION	text1_1CB	
PURPOSE	Callback function for the 'text1_1' text box ('Zero' filter range [%]) in FORM1	
SYNTAX	void text1_1CB(w, client_data, call_data)	
	Widget w	Parent widget ID
	CALIB *client_data;	A structure which contains the variables used to define the dialog list
	XtPointer call_data	Specialised structure with information useful to the function

INCLUDES	`#include <Xm/Xm.h>`
	`#include <Xm/Form.h>`
	`#include <Xm/Label.h>`
	`#include <Xm/PushB.h>`
	`#include <Xm/SelectionB.h>`
	`#include <Xm/List.h>`
	`#include <Xm/RowColumn.h>`
	`#include <Xm/ToggleB.h>`
	`#include <Xm/Frame.h>`
	`#include <Xm/Text.h>`
	`#include <Xm/Separator.h>`
	`#include <Xm/FileSB.h>`
	`#include <Xm/MessageB.h>`
	`#include "structure.h"`
	`#include "variables.h"`
	`#include "calib_function.h"`	

DESCRIPTION	Provides callback functionality to the 'text1_1' text box ('Zero' filter range [%]) in the GUI interface	

FUNCTION	text2CB	
PURPOSE	Callback function for the 'text2' text box in FORM3	
SYNTAX	`void text2CB(w, client_data, call_data)`	
	`Widget w`	Parent widget ID
	`CALIB *client_data;`	A structure which contains the variables used to define the dialog list
	`XtPointer call_data`	Specialised structure with information useful to the function

INCLUDES	`#include <Xm/Xm.h`
	`#include <Xm/Form.h`
	`#include <Xm/Label.h`
	`#include <Xm/PushB.h`
	`#include <Xm/SelectionB.h`
	`#include <Xm/List.h`
	`#include <Xm/RowColumn.h`
	`#include <Xm/ToggleB.h`
	`#include <Xm/Frame.h`
	`#include <Xm/Text.h`
	`#include <Xm/Separator.h`
	`#include <Xm/FileSB.h`
	`#include <Xm/MessageB.h`
	`#include "structure.h"`
	`#include "variables.h"`
	`#include "calib_function.h"`	

DESCRIPTION	Provides callback functionality to the 'text2' text box in the GUI interface	

FUNCTION	text3CB	
PURPOSE	Callback function for the 'text3' text box in FORM3	
SYNTAX	void text3CB(w, client_data, call_data)	
	Widget w	Parent widget ID
	CALIB *client_data;	A structure which contains the variables used to define the dialog list
	XtPointer call_data	Specialised structure with information useful to the function

INCLUDES	`#include <Xm/Xm.h`
	`#include <Xm/Form.h>`
	`#include <Xm/Label.h>`
	`#include <Xm/PushB.h>`
	`#include <Xm/SelectionB.h>`
	`#include <Xm/List.h>`
	`#include <Xm/RowColumn.h>`
	`#include <Xm/ToggleB.h>`
	`#include <Xm/Frame.h>`
	`#include <Xm/Text.h>`
	`#include <Xm/Separator.h>`
	`#include <Xm/FileSB.h>`
	`#include <Xm/MessageB.h>`
	`#include "structure.h"`
	`#include "variables.h"`
	`#include "calib_function.h"`	

DESCRIPTION	Provides callback functionality to the 'text3' text box in the GUI interface	

FUNCTION	toggle1_1	
PURPOSE	Callback function for the 'toggle1_1' toggle button (Activate 'zero' data points filter) in FORM1	
SYNTAX	`void toggle1_1(w, client_data, call_data)`	
	`Widget w`	Parent widget ID
	`CALIB *client_data;`	A structure which contains the variables used to define the dialog list
	`XmToggleButtonCallbackStruct *call_data`	Specialised structure with information useful to the function

INCLUDES	`#include <Xm/Xm.h>`
	`#include <Xm/Form.h>`
	`#include <Xm/Label.h>`
	`#include <Xm/PushB.h>`
	`#include <Xm/SelectionB.h>`
	`#include <Xm/List.h>`
	`#include <Xm/RowColumn.h>`
	`#include <Xm/ToggleB.h>`
	`#include <Xm/Frame.h>`
	`#include <Xm/Text.h>`
	`#include <Xm/Separator.h>`
	`#include <Xm/FileSB.h>`
	`#include <Xm/MessageB.h>`
	`#include "structure.h"`
	`#include "variables.h"`
	`#include "calib_function.h"`	

DESCRIPTION	Provides callback functionality to the 'toggle1_1' toggle button (Activate 'zero' data points filter) in the GUI interface	

FUNCTION	toggle7_1	
PURPOSE	Callback function for the toggle button 'toggle7_1' (Activate Chauvenet's Criterion) in FORM7	
SYNTAX	void toggle7_1(w, client_data, call_data)	
	Widget w	Parent widget ID
	CALIB *client_data;	A structure which contains the variables used to define the dialog list
	XmToggleButtonCallbackStruct *call_data	Specialised structure with information useful to the function
INCLUDES	#include <Xm/Xm.h #include <Xm/Form.h #include <Xm/Label.h #include <Xm/PushB.h #include <Xm/SelectionB.h #include <Xm/List.h #include <Xm/RowColumn.h #include <Xm/ToggleB.h #include <Xm/Frame.h #include <Xm/Text.h #include <Xm/Separator.h #include <Xm/FileSB.h #include <Xm/MessageB.h #include "structure.h" #include "variables.h" #include "calib_function.h"	
DESCRIPTION	Provides callback functionality to the toggle button 'toggle7_1' (Activate Chauvenet's Criterion) in the GUI interface	

FUNCTION	toggleCB	
PURPOSE	Callback function for the toggle button (Tolerances in R and H) in FORM3	
SYNTAX	void toggleCB(w, client_data, call_data)	
	Widget w	Parent widget ID
	CALIB *client_data;	A structure which contains the variables used to define the dialog list
	XmToggleButtonCallbackStruct *call_data	Specialised structure with information useful to the function
INCLUDES	#include <Xm/Xm.h #include <Xm/Form.h #include <Xm/Label.h #include <Xm/PushB.h #include <Xm/SelectionB.h #include <Xm/List.h #include <Xm/RowColumn.h #include <Xm/ToggleB.h #include <Xm/Frame.h #include <Xm/Text.h #include <Xm/Separator.h #include <Xm/FileSB.h #include <Xm/MessageB.h #include "structure.h" #include "variables.h" #include "calib_function.h"	
DESCRIPTION	Provides callback functionality to the toggle button (Tolerances in R and H) in the GUI interface	

FUNCTION	chauvenet	
PURPOSE	Performs Chauvenet's Criterion on a set of data	
SYNTAX	`void chauvenet(calibptr)`	
	`CALIB *calibptr`	Structure list which contains all variables used in CALIB

INCLUDES	`#include <stdio.h>`
	`#include <stdlib.h>`
	`#include <math.h>`
	`#include <Xm/Xm.h>`
	`#include "structure.h"`
	`#include "variables.h"`
	`#include "nrutil.h"`	

DESCRIPTION	Performs Chauvenet's Criterion on a set of data. A $N=5$ Chauvenet's Criterion is implemented in this function.	

FUNCTION	coeff	
PURPOSE	Generate an output file which contains the result obtained by CALIB	
SYNTAX	void coeff(calibptr)	
	CALIB *calibptr	Structure which contains all the variables used in CALIB

INCLUDES	`#include <stdio.h`
	`#include <stdlib.h`
	`#include <Xm/Xm.h`
	`#include "structure.h"`
	`#include "variables.h"`
	`#include "nrutil.h"`	

DESCRIPTION	Generate an output file that contains the result obtained by CALIB. The output file is in ASCII text format which contains the followings: • Balance calibration model • Balance calibration equation • Number of coefficients in the calibration equation • Mathematical regression model used • Tolerance in the original [H] measurements • Tolerance in the original [R] measurements • Original number of data set in the data input file • Number of data set used for optimised calibration matrix calculation • Number of data set eliminated by the optimisation routine • The kind of optimisation technique used in optimising the data (standard error optimisation / Chauvenet's Criterion optimisation) • Balance calibration coefficients matrix • Optimised calibration coefficients matrix • Normalised balance calibration matrix • Sensitivity matrix • Standard errors for each balance component • Multiple correlation for each component	

FUNCTION	convert5to6	
PURPOSE	Convert a 5 components coefficient matrix to a 6 components coefficient matrix	
SYNTAX	`void convert5to6(calibptr)`	
	`CALIB *calibptr`	Structure which contains all the variables used in CALIB

INCLUDES	`#include <stdio.h>`
	`#include <stdlib.h>`
	`#include <Xm/Xm.h>`
	`#include "structure.h"`
	`#include "variables.h"`
	`#include "nrutil.h"`	

DESCRIPTION	Convert a 5 components coefficient matrix to a 6 components coefficient matrix. This function enable CALIB to perform analysis on 5 component strain gauge balance, such as the one used in the transonic wind tunnel.	

FUNCTION	dataout	
PURPOSE	Output a text file which contains the calibration data set which will be used by CALIB	
SYNTAX	`void dataout(calibptr)`	
	`CALIB *calibptr`	Structure which contains all the variables used in CALIB

INCLUDES	`#include <stdio.h>`
	`#include <stdlib.h>`
	`#include <Xm/Xm.h>`
	`#include "structure.h"`
	`#include "variables.h"`	

DESCRIPTION	Output an ASCII text file that contains the calibration data set, which is read into the computer memory by the function *rfile(calibptr)*. The purpose of this function is to allow the end user to make sure what is being analysed by the program is what is in the original calibration data file.	

FUNCTION	filter	
PURPOSE	Performs zero filtering procedure to eliminate close to zero values from the original calibration data set.	
SYNTAX	`void filter(calibptr)`	
	`CALIB *calibptr`	Structure which contains all the variables used in CALIB

INCLUDES	`#include <stdio.h`
	`#include <stdlib.h`
	`#include <Xm/Xm.h`
	`#include "structure.h"`
	`#include "variables.h"`
	`#include "nrutil.h"`	

DESCRIPTION	Performs zero filtering procedure to eliminate close to zero values from the original calibration data set. The definition of close to zero value can be defined by the user during run-time through the GUI interface.	

FUNCTION	gaussj	
PURPOSE	Performs Gauss-Jordan elimination with full pivoting	
SYNTAX	`void gaussj(double **a, int n, double **b, int m)`	
	`double **a`	Input matrix
	`int n`	Number of row and column of the input matrix
	`double **b`	Output matrix (inverted matrix)
	`int m`	Number of right-hand side vectors

INCLUDES	`#include <stdio.h`
	`#include <math.h`
	`#include "nrutil.h"`	

DESCRIPTION	Performs Gauss-Jordan elimination with full pivoting. (Refer to Numerical Recipes in C, Chapter 2)	

155

FUNCTION	leastsquare1	
PURPOSE	Calculate the balance calibration coefficient matrix using the multivariable regression least squares method.	
SYNTAX	`void leastsquare1(calibptr)`	
	`CALIB *calibptr`	Structure which contains all the variables used in CALIB

INCLUDES	`#include <stdio.h`
	`#include <stdlib.h`
	`#include <stddef.h`
	`#include <Xm/Xm.h`
	`#include "structure.h"`
	`#include "variables.h"`
	`#include "nrutil.h"`	

DESCRIPTION	Calculate the balance calibration coefficient matrix using the multivariable regression least squares method.	

FUNCTION	leastsquare2_1	
PURPOSE	Calculate the balance calibration coefficient matrix for the calibration model: [H]=[C][R-H] using the Ramaswamy least squares method.	
SYNTAX	`void leastsquare2_1(calibptr)`	
	`CALIB *calibptr`	Structure which contains all the variables used in CALIB

INCLUDES	`#include <stdio.h`
	`#include <stdlib.h`
	`#include <math.h`
	`#include <Xm/Xm.h`
	`#include "structure.h"`
	`#include "variables.h"`
	`#include "nrutil.h"`	

DESCRIPTION	Calculate the balance calibration coefficient matrix for the calibration model: [H]=[C][R-H] using the Ramaswamy least squares method.	

FUNCTION	leastsquare2_2
PURPOSE	Calculate the balance calibration coefficient matrix for the calibration model: [R]=[C][H] using the Ramaswamy least squares method.
SYNTAX	`void leastsquare2_2(calibptr)`
	`CALIB *calibptr` — Structure which contains all the variables used in CALIB

INCLUDES	`#include <stdio.h`
	`#include <stdlib.h`
	`#include <math.h`
	`#include <Xm/Xm.h`
	`#include "structure.h"`
	`#include "variables.h"`
	`#include "nrutil.h"`

DESCRIPTION	Calculate the balance calibration coefficient matrix for the calibration model: [R]=[C][H] using the Ramaswamy least squares method.

FUNCTION	leastsquare2_3
PURPOSE	Calculate the balance calibration coefficient matrix for the calibration model: [H]=[C][R] using the Ramaswamy least squares method.
SYNTAX	`void leastsquare2_3(calibptr)`
	`CALIB *calibptr` — Structure which contains all the variables used in CALIB

INCLUDES	`#include <stdio.h`
	`#include <stdlib.h`
	`#include <math.h`
	`#include <Xm/Xm.h`
	`#include "structure.h"`
	`#include "variables.h"`
	`#include "nrutil.h"`

DESCRIPTION	Calculate the balance calibration coefficient matrix for the calibration model: [H]=[C][R] using the Ramaswamy least squares method.

FUNCTION	optimise	
PURPOSE	Eliminate any outliers defined by the user from the original calibration data set.	
SYNTAX	`void optimise(calibptr)`	
	`CALIB *calibptr`	Structure which contains all the variables used in CALIB

INCLUDES	`#include <stdio.h`
	`#include <stdlib.h`
	`#include <math.h`
	`#include <Xm/Xm.h`
	`#include "structure.h"`
	`#include "variables.h"`
	`#include "nrutil.h"`	

DESCRIPTION	Eliminate any outliers defined by the user from the original calibration data set. Outliers are defined by either the standard error optimisation techniques or the Chauvenet's Criterion optimisation technique.	

FUNCTION	rev_calib1	
PURPOSE	Performs reverse calibration on calibration model: [H]=[C][R-H].	
SYNTAX	`void rev_calib1(calibptr)`	
	`CALIB *calibptr`	Structure which contains all the variables used in CALIB

INCLUDES	`#include <stdio.h`
	`#include <stdlib.h`
	`#include <math.h`
	`#include <Xm/Xm.h`
	`#include "structure.h"`
	`#include "variables.h"`
	`#include "nrutil.h"`	

DESCRIPTION	Performs reverse calibration on calibration model: [H]=[C][R-H]. This function uses a iterative reverse calibration procedures to obtain the solutions.	

FUNCTION	rev_calib2	
PURPOSE	Performs reverse calibration on calibration model: [R]=[C][H].	
SYNTAX	void rev_calib2(calibptr)	
	CALIB *calibptr	Structure which contains all the variables used in CALIB
INCLUDES	#include <stdio.h> #include <stdlib.h> #include <math.h> #include <Xm/Xm.h> #include "structure.h" #include "variables.h" #include "nrutil.h"	
DESCRIPTION	Performs reverse calibration on calibration model: [R]=[C][H]. This function uses a iterative reverse calibration procedures to obtain the solutions.	

FUNCTION	rev_calib3	
PURPOSE	Performs reverse calibration on calibration model: [H]=[C][R].	
SYNTAX	void rev_calib13calibptr)	
	CALIB *calibptr	Structure which contains all the variables used in CALIB
INCLUDES	#include <stdio.h> #include <stdlib.h> #include <math.h> #include <Xm/Xm.h> #include "structure.h" #include "variables.h" #include "nrutil.h"	
DESCRIPTION	Performs reverse calibration on calibration model: [H]=[C][R]. This function uses a direct (non-iterative) reverse calibration procedures to obtain the solutions.	

159

FUNCTION	rfile	
PURPOSE	Reads the data input file and the balance load range file	
SYNTAX	`void rfile(calibptr)`	
	`CALIB *calibptr`	Structure which contains all the variables used in CALIB

INCLUDES	`#include <stdio.h`
	`#include <stdlib.h`
	`#include <stddef.h`
	`#include <Xm/Xm.h`
	`#include "structure.h"`
	`#include "variables.h"`
	`#include "nrutil.h"`	

DESCRIPTION	Reads the data input file and the balance load range file. It introduces a non-zero constant, called TINY (1e-307) to all data points equal to zero. This will eliminate any possibility of getting a floating point error (divided by zero) during run-time.	

FUNCTION	rfile2	
PURPOSE	Reads the balance design load range from the input data file (des_load.dat)	
SYNTAX	`void rfile2(calibptr)`	
	`CALIB *calibptr`	Structure which contains all the variables used in CALIB

INCLUDES	`#include <stdio.h`
	`#include <stdlib.h`
	`#include <stddef.h`
	`#include <Xm/Xm.h`
	`#include "structure.h"`
	`#include "variables.h"`
	`#include "nrutil.h"`	

DESCRIPTION	Reads the balance design load range from the input data file (des_load.dat).	

FUNCTION	RR_model1	
PURPOSE	Generates RR2 matrix for second and third order calibration equations for the calibration model: [H]=[C][R-H]	
SYNTAX	`void RR_model1(calibptr)`	
	`CALIB *calibptr`	Structure which contains all the variables used in CALIB

INCLUDES	`#include <stdio.h`
	`#include <stdlib.h`
	`#include <math.h`
	`#include <Xm/Xm.h`
	`#include "structure.h"`
	`#include "nrutil.h"`	

DESCRIPTION	Generates RR2 matrix for the second and third order calibration equations for the calibration model: [H]=[C][R-H]. The RR2 matrix is then use with the least squares regression methods to calculate the balance calibration coefficient matrix.	

FUNCTION	RR_model2	
PURPOSE	Generates RR2 matrix for second and third order calibration equations for the calibration model: [R]=[C][H]	
SYNTAX	`void RR_model2(calibptr)`	
	`CALIB *calibptr`	Structure which contains all the variables used in CALIB

INCLUDES	`#include <stdio.h`
	`#include <stdlib.h`
	`#include <math.h`
	`#include <Xm/Xm.h`
	`#include "structure.h"`
	`#include "nrutil.h"`	

DESCRIPTION	Generates RR2 matrix for the second and third order calibration equations for the calibration model: [R]=[C][H]. The RR2 matrix is then use with the least squares regression methods to calculate the balance calibration coefficient matrix.	

161

FUNCTION	RR_model3	
PURPOSE	Generates RR2 matrix for second and third order calibration equations for the calibration model: [H]=[C][R]	
SYNTAX	`void RR_model3(calibptr)`	
	`CALIB *calibptr`	Structure which contains all the variables used in CALIB

INCLUDES	`#include <stdio.h>`
	`#include <stdlib.h>`
	`#include <math.h>`
	`#include <Xm/Xm.h>`
	`#include "structure.h"`
	`#include "nrutil.h"`	

DESCRIPTION	Generates RR2 matrix for the second and third order calibration equations for the calibration model: [H]=[C][R]. The RR2 matrix is then use with the least squares regression methods to calculate the balance calibration coefficient matrix.	

FUNCTION	statistic	
PURPOSE	Calculates the standard errors and the coefficient of multiple correlation for each component of the strain gauge balance described by the balance calibration equation.	
SYNTAX	`void statistic(calibptr)`	
	`CALIB *calibptr`	Structure which contains all the variables used in CALIB

INCLUDES	`#include <stdio.h>`
	`#include <stdlib.h>`
	`#include <math.h>`
	`#include <Xm/Xm.h>`
	`#include "structure.h"`
	`#include "nrutil.h"`	

DESCRIPTION	Calculates the standard errors and the coefficient of multiple correlation for each component of the strain gauge balance described by the balance calibration equation.	

Appendix E: ACTUATOR Software Functions

E.1. ACTUATOR Software Function Index

FUNCTION	DESCRIPTION	FILE
Bittest	Bit test	act_cntrl.c
CalcLvdt	Converts an Angle to an LVDT reading	act_cntrl.c
CalcAngle	Converts LVDT reading to an angle	act_cntrl.c
CheckActData	Check for valid actuator data	act_cntrl.c
initAct	Reads actuator initialisation file and initialises variables	act_cntrl.c
SetUp	Initialises actuators	act_cntrl.c
CancelOpenCb	Cancel call back for open file window	actuator.c
Init_all	Initialising function for actuators	actuator.c
OpenFile	Create file selection dialog box	actuator.c
OpenFileCb	Open file call back	actuator.c
Quit	Quits program	actuator.c
Stop	Stops actuators	actuator.c
Update	Update display variables	actuator.c
ActCntrlMW_calib	Calibration selection	actuator_stubs.c
AcCntrlMW_ExitButton_CB1	Exit button call back	actuator_stubs.c
ActCntrlMW_MoveButton_CB1	Move actuators	actuator_stubs.c
ActCntrlMW_StopButton_CB1	Stop button call back (calibration window)	actuator_stubs.c
Cancel_button_CB1	'Cancel' call back in calibration window	actuator_stubs.c
CheckNumChar	Checks string for non-numeric characters	actuator_stubs.c
MinusButton_CB1	'-' button call back in calibration window	actuator_stubs.c
PlusButton_CB1	'+' button call back in calibration window	actuator_stubs.c
PulseButton_CB1	Pulse button call back (calibration window)	actuator_stubs.c
ReadButton_CB1	'Read' button call back in calibration window	actuator_stubs.c
RemoveDialButton	Remove button from window	actuator_stubs.c

Reset_ok_CB	?OK? button call back on ?Reset Warning? popup	actuator_stubs.c
Reset_warning	Create pop-up dialog asking if the user wishes to continue with the actuator reset	actuator_stubs.c
ResetButton_CB1	Reset button call back (main window)	actuator_stubs.c
RTSetbutt_no	Changes the button_no variable to the selected actuator channel	actuator_stubs.c
SecondW_Okbutton	'OK' call back in calibration window	actuator_stubs.c
set_warning	Prints warning message to screen (creates warning dialog window)	actuator_stubs.c
StopButton_CB2	Stop button call back (main window)	actuator_stubs.c
Warn_ok_CB1	'OK' button call back on warning window	actuator_stubs.c
button_create	Creates a button widget	actuator_ui.c
cntrlpane_create	Creates control pane widget	actuator_ui.c
label_create	Creates a label widget	actuator_ui.c
mainwindow_create	Creates main window widgets	actuator_ui.c
MainwindowInfo_clear	Clear all information in variable instance	actuator_ui.c
menubutton_create	Creates a pull down menu widget	actuator_ui.c
menubutton_menu_create	Creates the menu in a pull down menu widget	actuator_ui.c
MW_mainwindow_initialise	Initialise and create the main window	actuator_ui.c
MW_target_create	Create the target angle widget in the main window	actuator_ui.c
panedwin_create	Creates paned window widget	actuator_ui.c
secondwindow_create	Creates calibration window widgets	actuator_ui.c
Secondwindow_init	Initialise and create the main window	actuator_ui.c
SecondwindowInfo_Clear	Clear all information in variable instance	actuator_ui.c
TextBox_create	Creates an editable text box widget	actuator_ui.c
TextBox2_create	Creates a text box widget	actuator_ui.c
spline	Initialise cubic spline (calculate the second derivative for the tabulated data)	cub_spline.f
splint	Initialise cubic spline (calculate the second derivative for the tabulated data)	cub_spline.f

E.2. ACTUATOR Software Function Summaries

FUNCTION	Bittest	
PURPOSE	Bit test	
SYNTAX	`int bittest(unsigned short int data, int i);`	
	`unsigned short int data;`	The variable to be tested
	`int i`	The bit number to be tested
RETURN VALUES	Value of bit number i	

FUNCTION	CalcLvdt	
PURPOSE	Converts an Angle to an LVDT reading	
SYNTAX	`double CalcLvdt(double Angle, int i)`	
	`double Angle;`	Angle
	`int i;`	actuator channel
INCLUDES	`#include "act_cntrl.h"`	
RETURN VALUES	LVDT required for given angle	

FUNCTION	CalcAngle	
PURPOSE	Converts LVDT reading to an angle	
SYNTAX	`double CalcAngle(double lvdt, int i)`	
	`double lvdt;`	lvdt reading
	`int i;`	actuator channel
INCLUDES	`#include "act_cntrl.h"`	
RETURN VALUES	Angle obtained from given LVDT reading	

FUNCTION	CheckActData	
PURPOSE	Check for valid actuator data	
SYNTAX	`int CheckActData(ActDataPtr adp, double data, int i, int typ)`	
	`ActDataPtr adp;`	Pointer to actuator data structure
	`double data;`	Data to be checked
	`int i;`	actuator channel
	`int typ;`	type of data (1 if angle, 2 if LVDT)
INCLUDES	`#include "act_cntrl.h"`	
RETURN VALUES	-2 -1 0 1	Channel number not valid Cubic spline not initialised Data O.K. Data out of range

FUNCTION	initAct	
PURPOSE	Reads actuator initialisation file and initialises variables	
SYNTAX	`void initAct(char *filename, ActDataPtr adp);`	
	`char *filename;`	Name of actuator initialisation file
	`ActDataPtr adp;`	Pointer to actuator data structure
INCLUDES	`#include "act_cntrl.h"`	
DESCRIPTION	Main actuator initialisation function	

FUNCTION	SetUp	
PURPOSE	Initialises actuators	
SYNTAX	`int SetUp(char *warnlabel, ActDataPrt adp);`	
	`char *warnlabel;`	Warning string if error occurs
	`ActDataPtr adp;`	Pointer to actuator data structure
INCLUDES	`#include "act_cntrl.h"`	
DESCRIPTION	If an error occurs when initialising the actuators, the external variable Act_Err is set to 1.	
RETURN VALUES	0 1	Actuators initialised Actuators failed to initialised

FUNCTION	CancelOpenCb	
PURPOSE	Cancel call back for open file window	
SYNTAX	void CancelOpenCb(Widget w, XtPointer cd, XtPointer cb);	
	Widget w;	Parent widget
	XtPointer cd;	NULL
	XtPointer cb	NULL
DESCRIPTION	Call back for the cancel button	

FUNCTION	Init_all
PURPOSE	Initialising function
SYNTAX	void Init_all(void);
DESCRIPTION	This function is the main initialisation function. It opens communications with the actuators and the shared memory and initialises the windows

FUNCTION	OpenFile	
PURPOSE	Create file selection dialog box	
SYNTAX	Widget OpenFile(Widget w, int type);	
	Widget w;	Parent widget
	int type	Not used
RETURN VALUES	File selection dialog box widget	
DESCRIPTION	This dialog box is specifically designed to select the actuator initialisation file and is only called if the file name was not given when the program was executed.	

FUNCTION	OpenFileCb	
PURPOSE	Open file call back	
SYNTAX	`void OpenFileCb(Widget w, XtPointer cd, XtPointer cb);`	
	`Widget w;`	Parent widget
	`XtPointer cd;`	Variable type
	`XtPointer cb`	Pointer to the file selection call back structure
DESCRIPTION	Call back routine when a file is selected. Initialisation is initiated once file is selected.	

FUNCTION	Quit	
PURPOSE	Quits program	
SYNTAX	`void Quit(Widget w, XButtonEvent *event, String *args, int *n);`	
	`Widget w;`	
	`XbuttonEvent *event;`	
	`String *args;`	
	`int *n;`	
DESCRIPTION	This Quit function is activated using the keyboard override system and hence is a callback function with the required variables.	

FUNCTION	Stop	
PURPOSE	Stops actuators	
SYNTAX	`void Stop(Widget w, XButtonEvent *event, String *args, int *n);`	
	`Widget w;`	
	`XbuttonEvent *event;`	
	`String *args;`	
	`int *n;`	
DESCRIPTION	This Stop function is activated using the keyboard override system and hence is a callback function with the required variables.	

FUNCTION	Update	
PURPOSE	Update display variables	
SYNTAX	`void Update(XtPointer client_data, XtintervalId *id);`	
	`XtPointer client_data;`	
	`XtintervalId *id;`	
DESCRIPTION	Updates all actuator variables	

FUNCTION	ActCntrlMW_calib	
PURPOSE	Calibration selection	
SYNTAX	`void ActCntrlMW_calib(Widget w, XtPointer clientData, XtPointer callData);`	
	`Widget w;`	Parent widget
	`XtPointer clientData;`	Pointer to actuator data structure
	`XtPointer callData;`	Not used
INCLUDES	`#include "actuator_ui.h"`	
DESCRIPTION	Opens calibration (manual mode) window with the selected actuator channel. Also closes main window.	

FUNCTION	ActCntrlMW_ExitButton_CB1	
PURPOSE	Exit button call back	
SYNTAX	`void ActCntrlMW_ExitButton_CB1(Widget w, XtPointer clientData, XtPointer callData);`	
	`Widget w;`	Parent widget
	`XtPointer clientData;`	Not used
	`XtPointer callData;`	Not used
INCLUDES	`#include "actuator_ui.h"`	
DESCRIPTION	Checks for moving actuators before quitting the program. If moving the 'Stop' function is executed and an error message is displayed.	

FUNCTION	ActCntrlMW_MoveButton_CB1	
PURPOSE	Move actuators	
SYNTAX	void ActCntrlMW_MoveButton_CB1(Widget w, XtPointer clientData, XtPointer callData);	
	Widget w;	Parent widget
	XtPointer clientData;	Pointer to actuator data structure
	XtPointer callData;	Not used
INCLUDES	#include "actuator_ui.h"	
DESCRIPTION	Call back routine when move button in the main window is selected. This routine reads the target angles, converts them to required LVDT values for each channel, then sends the appropriate 'Move' command to the actuators.	

FUNCTION	ActCntrlMW_StopButton_CB1	
PURPOSE	Stop button call back (Calibration window)	
SYNTAX	void ActCntrlMW_StopButton_CB1(Widget w, XtPointer clientData, XtPointer callData);	
	Widget w;	Parent widget
	XtPointer clientData;	Not used
	XtPointer callData;	Not used
INCLUDES	#include "actuator_ui.h"	
DESCRIPTION	Sends manual stop command to actuators	

FUNCTION	Cancel_button_CB1	
PURPOSE	'Cancel' call back in calibration window	
SYNTAX	void Cancel_button_CB1(Widget w, XtPointer clientData, XtPointer callData);	
	Widget w;	Parent widget
	XtPointer clientData;	Pointer to actuator data structure
	XtPointer callData;	Not used
INCLUDES	#include "actuator_ui.h"	
DESCRIPTION	Closes calibration window and opens the main window without saving the new calibration data.	

FUNCTION	CheckNumChar	
PURPOSE	Checks string for non-numeric characters	
SYNTAX	`int CheckNumChar(char *Text)`	
	`char *Text`	String to check
INCLUDES	`#include "actuator_ui.h"`	
RETURN VALUES	-1 0 1	String contains only null characters String contains only numeric characters String contains non-numeric characters
DESCRIPTION	If string contains characters other than the numbers 0 to 9, '-', or '+' function will return a failed condition.	

FUNCTION	MinusButton_CB1	
PURPOSE	'-' button call back in calibration window	
SYNTAX	`void MinusButton_CB1(Widget w, XtPointer clientData,` `XtPointer callData);`	
	`Widget w;`	Parent widget
	`XtPointer clientData;`	Pointer to actuator data structure
	`XtPointer callData;`	Not used
INCLUDES	`#include "actuator_ui.h"`	
DESCRIPTION	Sends manual move command to actuator (negative direction is defined in the initialisation file). If the actuator is in 'continuous' mode all buttons except 'Stop' are made inactive.	

FUNCTION	PlusButton_CB1	
PURPOSE	'+' button call back in calibration window	
SYNTAX	`void PlusButton_CB1(Widget w, XtPointer clientData, XtPointer callData);`	
	`Widget w;`	Parent widget
	`XtPointer clientData;`	Pointer to actuator data structure
	`XtPointer callData;`	Not used
INCLUDES	`#include "actuator_ui.h"`	
DESCRIPTION	Sends manual move command to actuator (positive direction is defined in the initialisation file). If the actuator is in 'continuous' mode all buttons except 'Stop' are made inactive.	

FUNCTION	PulseButton_CB	
PURPOSE	Pulse button call back (calibration window)	
SYNTAX	`void PulseButton_CB(Widget w, XtPointer clientData, XtPointer callData);`	
	`Widget w;`	Parent widget
	`XtPointer clientData;`	Not used
	`XtPointer callData;`	Not used
INCLUDES	`#include "actuator_ui.h"`	
DESCRIPTION	Swaps between 'Pulse' mode and 'Continuous' mode while in the calibration mode	

FUNCTION	ReadButton_CB1	
PURPOSE	'Read' button call back in calibration window	
SYNTAX	`void ReadButton_CB1(Widget w, XtPointer clientData, XtPointer callData);`	
	`Widget w;`	Parent widget
	`XtPointer clientData;`	Pointer to actuator data structure
	`XtPointer callData;`	Not used
INCLUDES	`#include "actuator_ui.h"`	
DESCRIPTION	Reads the current LVDT and Target angle in the calibration window and writes the information to a temporary file for later saving.	

FUNCTION	RemoveDialButton	
PURPOSE	Remove button from window	
SYNTAX	`void RemoveDialButton(Widget w, int which_button);`	
	`Widget w;`	Parent widget
	`int which_button;`	Button to be removed
INCLUDES	`#include "actuator_ui.h"`	

FUNCTION	Reset_ok_CB	
PURPOSE	'OK' button call back on 'Reset Warning' popup	
SYNTAX	`void Reset_ok_CB(Widget w, XtPointer cd, XtPointer cb)`	
	`Widget w`	Parent widget
	`XtPointer cd`	Not used
	`XtPointer cb`	Not used
INCLUDES	`#include "actuator_ui.h"`	
DESCRIPTION	Closes 'Reset Warning' popup and continues with reset	

FUNCTION	Reset_warning
PURPOSE	Create pop-up dialog asking if the user wishes to continue with the actuator reset
SYNTAX	`void Reset_warning(Widget parent);`
	`Widget parent;` — Parent window
INCLUDES	`#include "actuator_ui.h"`

FUNCTION	ResetButton_CB1
PURPOSE	Reset button call back
SYNTAX	`void ResetButton_CB1(Widget w, XtPointer clientData,` `XtPointer callData);`
	`Widget w;` — Parent widget
	`XtPointer clientData;` — Not used
	`XtPointer callData;` — Not used
INCLUDES	`#include "actuator_ui.h"`
DESCRIPTION	Sends 'Reinitialise' command to actuators. Does a similar check to 'Exit' button call back

FUNCTION	RTSetbutt_no
PURPOSE	Changes the button_no variable to the selected actuator channel
SYNTAX	`void RTSetbutt_no(Widget widget, int i,` `XtPointer callData);`
	`Widget widget;` — Parent widget
	`int i;` — Selected actuator channel
	`XtPointer callData;` — Not used
INCLUDES	`#include "actuator_ui.h"`

FUNCTION	SecondW_Okbutton	
PURPOSE	'OK' call back in calibration window	
SYNTAX	`void SecondW_Okbutton(Widget w, XtPointer clientData,` `XtPointer callData);`	
	`Widget w;`	Parent widget
	`XtPointer clientData;`	Pointer to actuator data structure
	`XtPointer callData;`	Not used
INCLUDES	`#include "actuator_ui.h"`	
DESCRIPTION	Updates initialisation file then closes the calibration window and opens the main window	

FUNCTION	set_warning	
PURPOSE	Prints warning message to screen (creates warning dialog window)	
SYNTAX	`void set_warning(Widget parent, char *warnlabel);`	
	`Widget parent;`	Parent widget
	`char *warnlabel`	The string to be printed
INCLUDES	`#include "actuator_ui.h"`	

FUNCTION	StopButton_CB2	
PURPOSE	Stop button call back (Main window)	
SYNTAX	`void StopButton_CB2(Widget w, XtPointer clientData,` `XtPointer callData);`	
	`Widget w;`	Parent widget
	`XtPointer clientData;`	Not used
	`XtPointer callData;`	Not used
INCLUDES	`#include "actuator_ui.h"`	
DESCRIPTION	Stop actuators by sending Power Relay Control commands	

FUNCTION	Warn_ok_CB1	
PURPOSE	'OK' button call back on warning window	
SYNTAX	`void Warn_ok_CB1(Widget w, XtPointer cd, XtPointer cb);`	
	`Widget w;`	Parent widget
	`XtPointer cd;`	Not used
	`XtPointer cb;`	Not used
INCLUDES	`#include "actuator_ui.h"`	
DESCRIPTION	Closes the warning window	

FUNCTION	button_create	
PURPOSE	Creates a button widget	
SYNTAX	`Widget button_create(Widget parent, char *label, int Nx, int Ny);`	
	`Widget parent;`	parent widget
	`char *label;`	The string written on the button
	`int Nx;`	X position of widget
	`int Ny;`	Y position of widget
RETURN VALUES	Button widget	
DESCRIPTION	Variable label is the string written on the button, the widget's label is the variable label with the string "_button" appended to the end.	

FUNCTION	cntrlpane_create	
PURPOSE	Creates control pane widget	
SYNTAX	`Widget cntrlpane_create(Widget parent, char *Label);`	
	`Widget parent;`	parent widget
	`char *label;`	Control pane label
RETURN VALUES	Control pane widget	

FUNCTION	label_create	
PURPOSE	Creates a label widget	
SYNTAX	`Widget label_create(Widget parent, char *label, int Nx, int Ny);`	
	`Widget parent;`	parent widget
	`char *label;`	The string to be written
	`int Nx;`	X position of widget
	`int Ny;`	Y position of widget
RETURN VALUES	Label widget	
DESCRIPTION	Variable label is the string written to the screen, the widget's label is the variable label with the string "_label" appended to the end.	

FUNCTION	mainwindow_create	
PURPOSE	Creates main window widgets	
SYNTAX	`static int mainwindow_create(MainwindowInfo instance, Widget parent);`	
	`MainwindowInfo instance;`	Main window information structure
	`Widget parent;`	parent widget

FUNCTION	MainwindowInfo_Clear	
PURPOSE	Clear all information in variable instance	
SYNTAX	`int MainwindowInfo_Clear(MainwindowInfo instance);`	
	`MainwindowInfo instance`	Main window information structure

FUNCTION	menubutton_create	
PURPOSE	Creates a pull down menu widget	
SYNTAX	static int menubutton_create(MainwindowInfo instance, Widget parent, int Nx, int Ny);	
	MainwindowInfo instance;	Main window information structure
	Widget parent;	Parent widget
	int Nx;	X position of widget
	int Ny;	Y position of widget

FUNCTION	menubutton_menu_create	
PURPOSE	Creates the menu in a pull down menu widget	
SYNTAX	static int menubutton_menu_create(MainwindowInfo instance, Widget parent, ActDataPtr adp);	
	MainwindowInfo instance;	Main window information structure
	Widget parent;	Parent widget
	ActDataPtr adp;	Pointer to actuator data structure
DESCRIPTION	The elements of the menu is a list of the valid actuator channels	

FUNCTION	MW_mainwindow_initialise	
PURPOSE	Initialise and create the main window	
SYNTAX	int MW_mainwindow_initialise(MainwindowInfo instance, Widget parent, ActDataPtr adp);	
	MainwindowInfo instance;	Main window information structure
	Widget parent;	parent widget
	ActDataPtr adp;	Pointer to actuator data structure
INCLUDES	#include "actuator_ui.h"	
DESCRIPTION	Requires X-windows libraries	

FUNCTION	MW_target_create	
PURPOSE	Create the target angle widget in the main window	

SYNTAX	`static int`
	`MW_target_create(MainwindowInfo instance,`
	`Widget parent, char *label, int i, int j);`	

	`MainwindowInfo instance;`	Main window information structure
	`Widget parent;`	parent widget
	`char *label;`	Widget label
	`int i;`	Actuator channel number
	`int j;`	Position number
DESCRIPTION	Create a RowColumn widget consisting of a label widget and an editable Text widget. The X position of the widget is defined using XmATTACH_OPPOSITE_FORM and a right offset of -130. The Y position of the widget is based on the variable j	

FUNCTION	panedwin_create	
PURPOSE	Creates paned window widget	
SYNTAX	`Widget panedwin_create(Widget parent, char *Label);`	
	`Widget parent;`	parent widget
	`char *label;`	Paned window label
RETURN VALUES	Paned window widget	

FUNCTION	secondwindow_create	
PURPOSE	Creates calibration window widgets	

SYNTAX	`int`
	`secondwindow_create(SecondwindowInfo instance,`
	`Widget parent);`	

	`SecondwindowInfo instance;`	Second window information structure
	`Widget parent;`	parent widget

FUNCTION	Secondwindow_init	
PURPOSE	Initialise and create the main window	
SYNTAX	int Secondwindow_init(SecondwindowInfo instance, Widget parent, ActDataPtr adp);	
	SecondwindowInfo instance;	Second window information structure
	Widget parent;	parent widget
	ActDataPtr adp;	Pointer to actuator data structure
INCLUDES	#include "actuator_ui.h"	
DESCRIPTION	Requires X-windows libraries	

FUNCTION	SecondwindowInfo_Clear	
PURPOSE	Clear all information in variable instance	
SYNTAX	int SecondwindowInfo_Clear(SecondwindowInfo instance);	
	SecondwindowInfo instance	Calibration window information structure

FUNCTION	TextBox_create	
PURPOSE	Creates an editable text box widget	
SYNTAX	Widget TextBox2_create(Widget parent, char *label, int Nx, int Ny);	
	Widget parent;	parent widget
	char *label;	The string to be written
	int Nx;	X position of widget
	int Ny;	Y position of widget
RETURN VALUES	Text box widget	
DESCRIPTION	The length of the text box is 8 characters. This text box is not editable	

FUNCTION	TextBox2_create	
PURPOSE	Creates a text box widget	
SYNTAX	Widget TextBox_create(Widget parent, char *label, int Nx, int Ny);	
	Widget parent;	parent widget
	char *label;	The string to be written
	int Nx;	X position of widget
	int Ny;	Y position of widget
RETURN VALUES	Text box widget	
DESCRIPTION	The length of the text box is 8 characters. This text box is editable	

FUNCTION	Spline	
PURPOSE	Initialise cubic spline (calculate the second derivative for the tabulated data)	
SYNTAX	void spline(double x[], double y[], int n, double yp1, double ypn, double y2[]);	
	double x[], y[];	Tabulated array to interpolate
	int n;	number of elements in tabulated array
	double yp1;	first derivative at point 1
	double ypn;	first derivative at point n
	double y2[];	Tabulated second derivative
INCLUDE	#include "cub_spline.h"	
DESCRIPTION	This function is obtained from the book "Numerical Recipes in C". If yp1 and/or ypn are equal to 1 x 1030 or larger, the routine is signalled to set the corresponding boundary condition for a natural spline, with a zero second derivative on that boundary.	

FUNCTION	splint
PURPOSE	Initialise cubic spline (calculate the second derivative for the tabulated data)
SYNTAX	`void spline(double xa[], double ya[], double y2a[], int n, double x, double *y);`
	`double xa[], ya[];`	Tabulated array to interpolate
	`double y2a[];`	Tabulated second derivative (Calculated using function spline)
	`int n;`	Number of elements in array
	`double x`	Point at which to interpolate
	`double *y;`	Interpolated value
INCLUDE	`#include "cub_spline.h"`
DESCRIPTION	This function is obtained from the book "Numerical Recipes in C".

Appendix F: UDP Communication Software Functions

F.1. UDP Communication Software Function Index

FUNCTION	DESCRIPTION	FILE
clsLSWT_ModuleBcastDataBase	Shuts down the "Bcast" library code in an orderly fashion.	rdDataLib.c
getBcastDataStructurePointer	Provides fleeting but unfettered read access to all aspects of the latest Bcast packet captured for a given module.	rdDataLib.c
getRdBcastDataVerboseFlag	Allows the state of the verbose flag to be determined.	rdDataLib.c
opnLSWT_ModuleBcastDataBase	Initialises the "Bcast" library code so that broadcast data can be accessed on Bernoulli.	rdDataLib.c
rdAllLSWT_ModulesBcastPacketStats	Provides all statistics on Bcast packets with one function call.	rdDataLib.c
rdAllLSWT_ModulesBcastPacketCaptureCount	Provides latest Bcast packet capture count for all modules.	rdDataLib.c
rdAllLSWT_ModulesBcastPacketLossCount	Provides latest Bcast packet loss count for all modules.	rdDataLib.c
rdLSWT_ModuleBcastHeaderSequenceNumber	Provides latest Bcast header sequence number for a given module.	rdDataLib.c
rdLSWT_ModuleBcastPacketCaptureCount	Provides latest Bcast packet capture count for a given module.	rdDataLib.c

rdLSWT_ModuleBcastPacketLossCount	Provides latest Bcast packet loss count for a given module.	rdDataLib.c
rdLSWT_ModuleBcastStrings	Provides latest Bcast message strings for a given module.	rdDataLib.c
rdLSWT_ModuleBcastUnsignedShortInts	Provides latest Bcast data for a given module.	rdDataLib.c
rdLSWT_ModuleData	Provides latest Bcast data for a given module plus the packet count and loss statistics and the sequence number associated with that latest data.	rdDataLib.c
rdTotalLSWT_BcastPacketCaptureCount	Allows data transfer rate checking.	rdDataLib.c
setRdBcastDataVerboseFlag	Allows the state of the verbose flag to be changed at some point after initialisation of the library code.	rdDataLib.c

F.2. UDP Communication Software Function Summaries

FUNCTION	clsLSWT_ModuleBcastDataBase
PURPOSE	Shuts down the "Bcast" library code in an orderly fashion.
SYNTAX	`int clsLSWT_ModuleBcastDataBase ( void )`
INCLUDES	`#include rdDataLib.h`
RETURN VALUES	Returns an integer value of 0 (ie. Zero or False) if an error occurs, otherwise a non-zero is returned indicating a successful close operation.
DESCRIPTION	This function is the Bcast data access shut down routine. It closes access to the shared memory containing data sent from the LSWT modules via UDP packets to the testBcast daemon. The code checks that the global flag indicating that a previous open (see previously "opnLSWT_ModuleBcastDataBase()") was successful and if it was then it detaches the two Bcast shared memory areas from the program calling this close routine. If there has not been a previous successful open then an error message is output and the return value is set to zero. If the code is successful then it sets a global flag indicating that the Bcast module read data routines cannot be used and returns a non-zero value.

USEAGE	#include "rdDataLib.h"
	... Other includes ...
	if (! opnLSWT_ModuleBcastDataBase(0)) fprintf (stderr, "Open Failed\n");
	else {
	... Read data ...
	if (! clsLSWT_ModuleBcastDataBase()) fprintf (stderr, "Close Failed\n");
	... Do other clean ups and closures ...
	}

FUNCTION	getBcastDataStructurePointer	
PURPOSE	Provides fleeting but unfettered read access to all aspects of the latest Bcast packet captured for a given module.	
SYNTAX	`struct LSWT_UDP_INFO * getBcastDataStructurePointer ( int moduleID )`	
	`int moduleID;`	An integer that specifies the module[2].
INCLUDES	`#include rdDataLib.h`	
RETURN VALUES	Returns a NULL pointer if an error occurs, otherwise a pointer to the data structure is returned.	
DESCRIPTION	This low-level function makes possible read access to the latest Bcast packet captured for a given module. This function checks that the specified module is valid and then checks that data exists for the specified module and if so it then returns a pointer to that data to the calling program. There is no check to see that the open function has been called and this function does not guarantee that data pointed at will remain either static or up to date. Indeed the data accessed via this pointer is likely to become stale very quickly and be overwritten by a data packet from any module shortly after it becomes stale. If there is an invalid module specified or there is no data available then an error message is output and the return pointer is set to NULL.	

---

[2] Freestream Parameters Module = 1
Paramater Display Module = 2
Auxiliary Data Module = 3
Sting Column Rig Module = 4
Serial Hub Module = 5
AC Strain Gauge 1 Module = 7
AC Strain Gauge 2 Module = 8
DC Strain Gauge 1 Module = 9
DC Strain Gauge 2 Module = 10
Inclinometer Module = 11
Model Attitude Module = 12
Actuator Module = 13
Balance Calibration Module = 14

USEAGE	
| | ```
#include "rdDataLib.h"
... Other includes ...
struct LSWT_UDP_INFO * ptr;
... initialize other variables ...
if ( ! opnLSWT_ModuleBcastDataBase( 0 )) fprintf ( stderr, "Open
Failed\n" );
else {
  ptr = getBcastDataStructurePointer ( 12 );
  if ( ptr == NULL )
    fprintf ( stderr, "Unable to get pointer to module 12 data\n" );
  else {
    printf ( "Module 12 latest packet sequence field is %x\n",
      ptr-sequenceNumber );
    printf ( "Module 12 latest packet time field is %d\n",
      ( int ) ptr-timeSinceMidnightInMilliseconds );
  }
}
``` |

FUNCTION	**getRdBcastDataVerboseFlag**
PURPOSE	Allows the state of the verbose flag to be determined.
SYNTAX	`int getRdBcastDataVerboseFlag ( void );`
INCLUDES	`#include rdDataLib.h`
RETURN VALUES	Returns an integer value of 0 (ie. Zero or False or Off) when the verbose flag is false, otherwise a non-zero is returned indicating the flag is True or On.
DESCRIPTION	This function returns the state of the Bcast data library global verbose text output flag.
USEAGE	#include "rdDataLib.h" ... Other includes ... printf ("Verbose Flag is %s\n", (getRdBcastDataVerboseFlag () ? "On" : "Off")); ... Other statements ...

FUNCTION	opnLSWT_ModuleBcastDataBase	
PURPOSE	Initialises the "Bcast" library code so that broadcast data can be accessed on Bernoulli.	
SYNTAX	`int opnLSWT_ModuleBcastDataBase ( int verboseFlag );`	
	`int verboseFlag;`	Used to set a global flag to indicate if verbose message output is required. If it is set to zero then error messages are the only text output from this and other Bcast data access functions.
INCLUDES	`#include rdDataLib.h`	
RETURN VALUES	Returns an integer value of 0 (ie. Zero or False) if an error occurs, otherwise a non-zero is returned indicating a successful open operation.	
DESCRIPTION	This function is the Bcast data access initialisation routine. It opens access to the shared memory containing data sent from LSWT modules via UDP packets to the testBcast daemon. The code copies the "verboseFlag" parameter to a global flag variable then attaches the two Bcast shared memory areas and the Bcast semaphore to the program calling this open routine. If the verbose flag is on and the function is successful then the pointer values to the shared memory are printed out. If a shared memory cannot be attached or the semaphore is not available then an error message is output and the return value is set to zero. If the code is successful then it sets a global flag indicating that the other Bcast module data routines can be used and returns a non-zero value.	

USEAGE	`#include "rdDataLib.h"`
	... Other includes ...
	`int flag = 1;`
	... Do other initialisation ...
	`if ( ! opnLSWT_ModuleBcastDataBase( flag )) fprintf ( stderr, "Open Failed\n" );`
	`else {`
	... Read data ...
	`if ( ! clsLSWT_ModuleBcastDataBase()) fprintf ( stderr, "Close Failed\n" );`
	... Do other clean ups and closures ...
	`}`	

FUNCTION	rdAllLSWT_ModulesBcastPacketStats	
PURPOSE	Provides all statistics on Bcast packets with one function call.	
SYNTAX	`int rdAllLSWT_ModulesBcastPacketStats ( unsigned int * countPtr, unsigned int * lossPtr)`	
	`unsigned int * countPtr;`	Pointer to an array into which to put the unsigned integer packet capture count data for each module. An array size defined by LSWT_DUMMY_FOR_LAST_MODULE is expected. This pointer must not be a NULL pointer.
	`Unsigned int * lossPtr;`	Pointer to an array into which to put the unsigned integer packet loss data for each module. An array size defined by LSWT_DUMMY_FOR_LAST_MODULE is expected. This pointer must not be a NULL pointer.
INCLUDES	`#include rdDataLib.h`	
RETURN VALUES	Returns an integer value of 0 (ie. Zero or False) if an error occurs, otherwise a non-zero is returned indicating a successful read operation.	
DESCRIPTION	This function gives access to both the packet capture count and packet loss count statistics for all individual modules plus the total for all modules combined. This function checks that there has been a successful open (see previously "opnLSWT_ModuleBcastDataBase()") at some point in the past and if so it copies the latest available unsigned short integer statistics data from the shared memory to the calling programs memory. If there has not been a previous successful open then an error message is output and the return value is set to zero. If the code is successful then it returns a non-zero value.	

| USEAGE | ```
#include "rdDataLib.h"
... Other includes ...
int cnt;
unsigned int cntBfr[ LSWT_DUMMY_FOR_LAST_MODULE ];
unsigned int lossBfr[ LSWT_DUMMY_FOR_LAST_MODULE ];
... initialize other variables ...
if ( ! opnLSWT_ModuleBcastDataBase( 0 )) fprintf ( stderr, "Open
Failed\n" );
else {
  if ( ! rdAllLSWT_ModulesBcastPacketStats ( cntBfr, lossBfr ))
    fprintf ( stderr, "Unable to Read All Modules Packet Statistics\n"
);
  else {
    printf ( "Total Bcast Packets captured is %u, and lost is %u\n",
    cntBfr[ 0 ], lossBfr[ 0 ] );
    for( cnt = 1; cnt < LSWT_DUMMY_FOR_LAST_MODULE; cnt++
)
      printf ( "Module %d - %u captured %u lost\n",
           cnt, cntBfr[ cnt ], lossBfr[ cnt] );
  }
}
``` |
|---|---|

FUNCTION	rdAllLSWT_ModulesBcastPacketCaptureCount	
PURPOSE	Provides latest Bcast packet capture count for all modules.	
SYNTAX	`int rdAllLSWT_ModulesBcastPacketCaptureCount (` `unsigned int * countArrayPtr )`	
	`unsigned int *` `countArrayPtr;`	Pointer to an array of unsigned integers into which to copy all the latest packet capture counts. An array size defined by LSWT_DUMMY_FOR_LAST_MODULE is expected. This pointer must not be a NULL pointer.
INCLUDES	`#include rdDataLib.h`	
RETURN VALUES	Returns an integer value of 0 (ie. Zero or False) if an error occurs, otherwise a non-zero is returned indicating a successful read operation.	
DESCRIPTION	This function gives access to the latest Bcast packet capture counts for all modules. This function checks that there has been a successful open (see previously "opnLSWT_ModuleBcastDataBase()") at some point in the past and if so it copies the unsigned integer packet capture count array from the shared memory to the calling programs memory. If there has not been a previous successful open then an error message is output and the return value is set to zero. If the code is successful then it returns a non-zero value.	

USEAGE	`#include "rdDataLib.h"`
	… Other includes …
	`int cnt;`
	`unsigned int counts[ LSWT_DUMMY_FOR_LAST_MODULE ];`
	… initialize other variables …
	`if ( ! opnLSWT_ModuleBcastDataBase( 0 )) fprintf ( stderr, "Open`
	`Failed\n" );`
	`else {`
	`  if ( ! rdAllLSWT_ModulesBcastPacketCaptureCount ( counts ))`
	`    fprintf ( stderr, "Unable to Read all packet capture counts\n" );`
	`  else {`
	`    printf ( "The total packets captured is %u\n", counts[ 0 ] );`
	`    for ( cnt =1; cnt < LSWT_DUMMY_FOR_LAST_MODULE; cnt++`
	`)`
	`      printf( "%u packets captured from module %d\n, counts[ cnt ],`
	`cnt );`
	`  }`
	`}`	

191

FUNCTION	rdAllLSWT_ModulesBcastPacketLossCount	
PURPOSE	Provides latest Bcast packet loss count for all modules.	
SYNTAX	`int rdAllLSWT_ModulesBcastPacketLossCount ( unsigned int * lossArrayPtr )`	
	`unsigned int * lossArrayPtr;`	Pointer to an array of unsigned integers into which to copy all the latest packet loss counts. An array size defined by LSWT_DUMMY_FOR_LAST_MODULE is expected. This pointer must not be a NULL pointer.
INCLUDES	`#include rdDataLib.h`	
RETURN VALUES	Returns an integer value of 0 (ie. Zero or False) if an error occurs, otherwise a non-zero is returned indicating a successful read operation.	
DESCRIPTION	This function gives access to the latest Bcast packet loss counts for all modules. This function checks that there has been a successful open (see previously "opnLSWT_ModuleBcastDataBase()") at some point in the past and if so it copies the unsigned integer packet loss count array from the shared memory to the calling programs memory. If there has not been a previous successful open then an error message is output and the return value is set to zero. If the code is successful then it returns a non-zero value.	

USEAGE	`#include "rdDataLib.h"`
	… Other includes …
	`int cnt;`
	`unsigned int lossCnts[ LSWT_DUMMY_FOR_LAST_MODULE ];`
	… initialize other variables …
	`if ( ! opnLSWT_ModuleBcastDataBase( 0 )) fprintf ( stderr, "Open Failed\n" );`
	`else {`
	`  if ( ! rdAllLSWT_ModulesBcastPacketLossCount ( lossCnts ))`
	`    fprintf ( stderr, "Unable to Read all packet loss counts\n" );`
	`  else {`
	`    printf ( "The total packet loss count is %u\n", lossCnts[ 0 ] );`
	`    for ( cnt =1; cnt < LSWT_DUMMY_FOR_LAST_MODULE; cnt++ )`
	`      printf( "%u packets lost from module %d\n, lossCnts[ cnt ], cnt );`
	`  }`
	`}`	

FUNCTION	rdLSWT_ModuleBcastHeaderSequenceNumber	
PURPOSE	Provides latest Bcast header sequence number for a given module.	
SYNTAX	`int rdLSWT_ModuleBcastHeaderSequenceNumber ( int module, unsigned short int * sequencePtr )`	
	`int module;`	The integer identifier of the module that data is required from.
	`Unsigned short int * sequencePtr;`	Pointer to an unsigned short integer into which to copy the sequence number from the latest packet header. It must not be a NULL pointer.
INCLUDES	`#include rdDataLib.h`	
RETURN VALUES	Returns an integer value of 0 (ie. Zero or False) if an error occurs, otherwise a non-zero is returned indicating a successful read operation.	
DESCRIPTION	This function gives access to the latest Bcast packet header sequence from a particular module. This function first checks that the module number is valid, then it checks that there has been a successful open (see previously "opnLSWT_ModuleBcastDataBase()") at some time in the past and finally it checks that data is available for the specified module. If these conditions are met then it copies the unsigned short integer sequence value for the specified module from the shared memory to the calling programs memory. If the checks mentioned fail then an error message is output and the return value is set to zero. If the code is successful then it returns a non-zero value.	

USEAGE	`#include "rdDataLib.h"`
	… Other includes …
	int mod = 7;
	unsigned short int seqNum;
	… initialize other variables …
	if (! opnLSWT_ModuleBcastDataBase(0)) fprintf (stderr, "Open Failed\n");
	else {
	  if (! rdLSWT_ModuleBcastHeaderSequenceNumber (mod, &seqNum))
	    fprintf (stderr, "Unable to Read Module %d sequence number\n", mod);
	  else {
	    printf ("Module %d sequence number is %x\n", mod, seqNum);
	  }
	}	

FUNCTION	rdLSWT_ModuleBcastPacketCaptureCount	
PURPOSE	Provides latest Bcast packet capture count for a given module.	
SYNTAX	`int rdLSWT_ModuleBcastPacketCaptureCount ( int module, unsigned int * countPtr )`	
	`Int module;`	The integer identifier of the module that the capture count is required for.
	`Unsigned int * countPtr;`	A Pointer to an unsigned integer into which to copy the module's latest packet capture count. It must not be a NULL pointer.
INCLUDES	`#include RdDataLib.h`	
RETURN VALUES	Returns an integer value of 0 (ie. Zero or False) if an error occurs, otherwise a non-zero is returned indicating a successful read operation.	
DESCRIPTION	This function gives access to the latest Bcast packet capture count from a particular module. This function checks that the module number is valid and that there has been a successful open (see previously "opnLSWT_ModuleBcastDataBase()") at some point in the past and if so it copies the unsigned integer packet capture count for the specified module from the shared memory to the calling programs memory. If there has not been a previous successful open or the module value is invalid then an error message is output and the return value is set to zero. If the code is successful then it returns a non-zero value.	

USEAGE	`#include "rdDataLib.h"`
	… Other includes …
	`int mod =3;`
	`unsigned int count;`
	… initialize other variables …
	`if ( ! opnLSWT_ModuleBcastDataBase( 0 )) fprintf ( stderr, "Open Failed\n" );`
	`else {`
	`  if ( ! rdLSWT_ModuleBcastPacketCaptureCount ( mod, &count ))`
	`    fprintf ( stderr, "Unable to Read Module %d packet capture count\n", mod );`
	`  else {`
	`    printf ( "%u packets received from module %d \n", count, mod );`
	`  }`
	`}`	

194

FUNCTION	rdLSWT_ModuleBcastPacketLossCount
PURPOSE	Provides latest Bcast packet loss count for a given module.
SYNTAX	int rdLSWT_ModuleBcastPacketLossCount (int module, unsigned int * lossCntPtr)
	int module;	The integer identifier of the module that the loss count is required for.
	unsigned int * lossCntPtr;	A Pointer to an unsigned integer into which to copy the module's latest packet loss count. It must not be a NULL pointer.
INCLUDES	#include rdDataLib.h
RETURN VALUES	Returns an integer value of 0 (ie. Zero or False) if an error occurs, otherwise a non-zero is returned indicating a successful read operation.
DESCRIPTION	This function gives access to the latest Bcast packet loss count from a particular module. This function first checks that the module number is valid and that there has been a successful open (see previously "opnLSWT_ModuleBcastDataBase()") at some point in the past and if so it copies the unsigned integer packet loss count for the specified module from the shared memory to the calling programs memory. If there has not been a previous successful open or the module value is invalid then an error message is output and the return value is set to zero. If the code is successful then it returns a non-zero value.

USEAGE	#include "rdDataLib.h"
	… Other includes …
	int mod =11;
	unsigned int lossCnt;
	… initialize other variables …
	if (! opnLSWT_ModuleBcastDataBase(0)) fprintf (stderr, "Open Failed\n");
	else {
	  if (! rdLSWT_ModuleBcastPacketLossCount (mod, &lossCnt))
	    fprintf (stderr, "Unable to Read Module %d packet loss count\n", mod);
	  else {
	    printf ("Module %d packet loss count is %u\n", mod, lossCnt);
	  }
	}

FUNCTION	rdLSWT_ModuleBcastStrings	
PURPOSE	Provides latest Bcast message strings for a given module.	
SYNTAX	`int rdLSWT_ModuleBcastStrings ( int module, int vector, char * stringBuffer, int bufferSize )`	
	`int module;`	The integer identifier of the module that data is required from.
	`int vector;`	The integer vector number that selects the required message string.
	`char * stringBuffer;`	Pointer to a target buffer into which to put the message string. It must not be a NULL pointer.
	`int bufferSize;`	The integer that is the size of the buffer that the message string will be placed in. This integer should be greater than zero.
INCLUDES	`#include rdDataLib.h`	
RETURN VALUES	Returns an integer value of 0 (ie. Zero or False) if an error occurs, otherwise a non-zero is returned indicating a successful read operation.	
DESCRIPTION	This function gives access to Bcast message strings from a particular module. This function first checks that the module number is valid and that the vector specifies a read, then it checks that there has been a successful open (see previously "opnLSWT_ModuleBcastDataBase()") at some time in the past, then it checks that the buffer size is greater than zero, then it checks that vector 0x62 has been specified and finally it checks that data is available for the specified module. If all these conditions are met then it copies the latest available character string for the specified module from the shared memory to the calling programs memory. The number of characters copied is the smaller of the actual string length and the size of the buffer available to store the string, represented by bufferSize. Currently only one string is available in shared memory and this is the module identification string (vector 0x62). If any of the checks mentioned fail then an error message is output and the return value is set to zero. If the code is successful then it returns a non-zero value.	

| USEAGE | ```
#include "rdDataLib.h"
... Other includes ...
int mod = 1, vec = 0x62;
char str[ 16 ];
... initialize other variables ...
if ( ! opnLSWT_ModuleBcastDataBase( 0 )) fprintf ( stderr, "Open
Failed\n" );
else {
  if ( ! rdLSWT_ModuleBcastStrings ( mod, vec, str, sizeof( str )))
    fprintf ( stderr, "Unable to Read Module ID string\n" );
  else {
    printf ( "Module ID string is '%s'\n", str );
  }
}
``` |
|--------|------------------|

FUNCTION	rdLSWT_ModuleBcastUnsignedShortInts	
PURPOSE	Provides latest Bcast data for a given module.	
SYNTAX	`int rdLSWT_ModuleBcastUnsignedShortInts ( int module, int vector, unsigned short int * dataBffr, int * valuesInBffrPtr)`	
	`int module;`	The integer identifier of the module that data is required from.
	`Int vector;`	The integer vector number that selects the start of the required subset of data.
	`Unsigned short int * dataBffr;`	Pointer to a target buffer into which to put the unsigned short integer data. It must not be a NULL pointer.
	`Int * valuesInBffrPtr;`	Pointer to an integer that is the desired number of unsigned short integer data values to copy on entry and on exit is the actual number of data values in the data buffer. This pointer must not be a NULL pointer and the integer it points to on entry must be greater than zero.
INCLUDES	`#include rdDataLib.h`	
RETURN VALUES	Returns an integer value of 0 (ie. Zero or False) if an error occurs, otherwise a non-zero is returned indicating a successful read operation.	
DESC.	This function gives access to a subset of the Bcast data from a particular module. This function first checks that the module number is valid and that the vector specifies a read, then it checks that there has been a successful open (see previously "opnLSWT_ModuleBcastDataBase()") at some time in the past, then it checks that the number of values to copy is greater than zero, then it checks that the vector number is valid and finally it checks that data is available for the specified module. If all these conditions are met then it copies the latest available unsigned short integer data values for the specified module from the shared memory to the calling programs memory. The number of data values copied is the smaller of the desired value pointed at by valuesInBffrPtr and the available data values including and following the vector number. The value pointed at by valuesInBffrPtr is updated to reflect the actual number of unsigned short integer values copied out of shared memory. If any of the checks mentioned fail then an error message is output and the return value is set to zero. If the code is successful then it returns a non-zero value.	

| USEAGE | ```
#include "rdDataLib.h"
... Other includes ...
int cnt, dataCnt = 16;
int mod = 1, vec = 0x80;
unsigned short int dataBfr[ 16 ];
... initialize other variables ...
if ( ! opnLSWT_ModuleBcastDataBase( 0 )) fprintf ( stderr, "Open
Failed\n" );
else {
  if ( ! rdLSWT_ModuleBcastUnsignedShortInts ( mod, vec, dataBfr,
&dataCnt ))
    fprintf ( stderr, "Unable to Read Unsigned Integer Data\n" );
  else {
    for( cnt = 0; cnt < dataCnt; cnt++ ) printf ( "%x\n", dataBfr[ cnt ] );
  }
}
``` |
|---|---|

FUNCTION	rdLSWT_ModuleData	
PURPOSE	Provides latest Bcast data for a given module plus the packet count and loss statistics and the sequence number associated with that latest data.	
SYNTAX	`int rdLSWT_ModuleData ( int module, int vector, unsigned short int * dataBffr, int * valuesInBffrPtr, unsigned int * packetCaptureCountPtr, unsigned int * packetLossCountPtr, unsigned short int * packetSequenceNumPtr )`	
	`int module;`	The integer identifier of the module that data is required from.
	`int vector;`	The integer vector number that selects the required subset of data.
	`unsigned short int * dataBffr;`	Pointer to a target buffer into which to put the unsigned short integer data. It must not be a NULL pointer.
	`int * valuesInBffrPtr;`	Pointer to an integer that is the desired number of unsigned short integer data values to copy on entry and on exit is the actual number of data values in the data buffer. This pointer must not be a NULL pointer and the integer it points to on entry must be greater than zero.
	`unsigned int * packetCaptureCountPtr;`	Pointer to an unsigned integer that will be updated with the current packet capture count for the module. This pointer must not be a NULL pointer.
	`unsigned int * packetLossCountPtr;`	Pointer to an unsigned integer that will be updated with the current packet loss count for the module. This pointer must not be a NULL pointer.
	`unsigned short int * packetSequenceNumPtr;`	Pointer to an unsigned short integer that will be updated with the current packet sequence number for the module. This pointer must not be a NULL pointer.
INCLUDES	`#include rdDataLib.h`	

RETURN VALUES	Returns an integer value of 0 (ie. Zero or False) if an error occurs, otherwise a non-zero is returned indicating a successful read operation.
DESCRIPTION	This function gives access to a subset of the Bcast data plus additional information for a given module. This function first checks that the module number is valid and that the vector specifies a read, then it checks that there has been a successful open (see previously "opnLSWT_ModuleBcastDataBase()") at some time in the past, then it checks that the buffer size is greater than zero and finally it checks that data is available for the specified module. If all these conditions are met it copies the latest available unsigned short integer data values and additional information for the specified module from the shared memory to the calling programs memory. The number of data values copied is the smaller of the desired value pointed at by valuesInBffrPtr and the available data values including and following the vector number. The value pointed at by valuesInBffrPtr is updated to reflect the actual number of unsigned short integer values copied out of shared memory. If any of the checks fail then an error message is output and the return value is set to zero. If the code is successful then it returns a non-zero value.

USEAGE	`#include "rdDataLib.h"`
	... Other includes ...
	int mod = 1, vec = 0xf0;
	int dataCnt = 1;
	unsigned short int dataBfr[4];
	unsigned short int seq;
	unsigned int rec, lost;
	... Initialize other variables ...
	if (! opnLSWT_ModuleBcastDataBase(0)) fprintf (stderr, "Open Failed\n");
	else {
	  if (! rdLSWT_ModuleData (mod, vec, dataBfr, &dataCnt, &rec, &lost, &seq))
	    fprintf (stderr, "Unable to read module %d vector %x data\n", mod, vec);
	  else {
	    if (dataCnt 0)
	      printf ("Module %d, vector %d data is %x\n", mod, vec, dataBfr[0]);
	    printf ("%u packets have been captured for this module\n", rec);
	    printf ("%u packets have been lost from this module\n", lost);
	    printf ("The current module %d sequence number is %x\n", mod, seq);
	    }
	}

FUNCTION	rdTotalLSWT_BcastPacketCaptureCount
PURPOSE	Allows data transfer rate checking.
SYNTAX	`int rdTotalLSWT_BcastPacketCaptureCount ( unsigned int * countPtr)`
	`unsigned int * countPtr;`	Pointer to an unsigned integer into which to put the current total number of packets processed by the testBcast daemon. It must not be a NULL pointer.
INCLUDES	`#include rdDataLib.h`
RETURN VALUES	Returns an integer value of 0 (ie. Zero or False) if an error occurs, otherwise a non-zero is returned indicating a successful read operation.
DESCRIPTION	This function gives access to the latest Bcast total packet count. The total packet count is incremented by one each time a packet from any module is written to shared memory. This function checks that there has been a successful open (see previously "opnLSWT_ModuleBcastDataBase()") at some time in the past and if so it then copies the total packet count from shared memory to the calling programs memory. If there has not been a previous successful open then an error message is output and the return value is set to zero. If the code is successful then it returns a non-zero value.

USEAGE	`#include "rdDataLib.h"`
	… Other includes …
	`unsigned int cnt;`
	… initialize other variables …
	`if ( ! opnLSWT_ModuleBcastDataBase( 0 )) fprintf ( stderr, "Open Failed\n" );`
	`else {`
	`  if ( ! rdTotalLSWT_BcastPacketCaptureCount ( &cnt))`
	`    fprintf ( stderr, "Unable to get Total Packet Capture Count\n" );`
	`  else {`
	`    printf( "%u Packets captured so far!\n", cnt );`
	`  }`
	`}`

FUNCTION	setRdBcastDataVerboseFlag
PURPOSE	Allows the state of the verbose flag to be changed at some point after initialisation of the library code.
SYNTAX	`void setRdBcastDataVerboseFlag ( int flag );`
	`int flag;` Used to set the global verbose flag to indicate if verbose message output is required. If it is set to zero then error messages are the only text output from this and other Bcast data access functions.
INCLUDES	`#include rdDataLib.h`
RETURN VALUES	None.
DESCRIPTION	This function sets the state of the Bcast data library verbose text output flag. The code sets the global verbose text output flag equal to the result of comparing the flag parameter as not equal with false (ie. Zero).

USEAGE	`#include "rdDataLib.h"`
	… Other includes …
	int flag = 1;
	… Do other initialization …
	if (! opnLSWT_ModuleDataBase(flag)) fprintf (stderr, "Open Failed\n");
	else {
	  flag = 0;
	  setRdBcastDataVerboseFlag(flag);
	  … Read Data (without any verbose messages) …
	}

Appendix G: VXI Communication Software Functions

G.1. VXI Communication Software Function Index

FUNCTION	DESCRIPTION	FILE
clsLSWT_VxiModuleDataBase	Shuts down the VXI library code in an orderly fashion.	rdDataLib.c
getVxiDataLibVerboseFlag	Allows the state of the verbose flag to be determined.	rdDataLib.c
opnLSWT_VxiModuleDataBase	Initialises the VXI library code so that VXI data can be accessed on the LSWT host computer Bernoulli.	rdDataLib.c
rdLSWT_VxiDataLinesProcessed	Allows data transfer rate checking.	rdDataLib.c
rdLSWT_VxiModuleData	Allow access to latest VXI data plus the time stamp and total number of data values associated with that latest data.	rdDataLib.c
rdLSWT_VxiModuleDoubles	Allow access to latest VXI data.	rdDataLib.c
setVxiDataLibVerboseFlag	Allows the state of the verbose flag to be changed at some point after initialisation of the library code.	rdDataLib.c

G.2. VXI Communication Software Function Summaries

FUNCTION	clsLSWT_VxiModuleDataBase
PURPOSE	Shuts down the VXI library code in an orderly fashion.
SYNTAX	`int clsLSWT_VxiModuleDataBase ( void )`
INCLUDES	`#include rdDataLib.h`
RETURN VALUES	Returns an integer value of 0 (ie. Zero or False) if an error occurs, otherwise a non-zero is returned indicating a successful close operation.
DESCRIPTION	This function is the VXI data access shut down routine. It closes access to the shared memory containing data sent from the VXI module through HP Vee and the vxiDataD daemon. The code checks that the global flag indicating that a previous open (see previously "opnLSWT_VxiModuleDataBase()") was successful and if it was then it detaches the two VXI shared memory areas from the program calling this close routine. If there has not been a previous successful open then an error message is output and the return value is set to zero. If the code is successful then it sets a global flag indicating that the VXI module read data routines cannot be used and returns a non-zero value.

USEAGE	`#include "rdDataLib.h"`
	... Other includes ...
	if (! opnLSWT_VxiModuleDataBase(0)) fprintf (stderr, "Open Failed\n");
	else {
	   ... Read data ...
	  if (! clsLSWT_VxiModuleDataBase()) fprintf (stderr, "Close Failed\n");
	   ... Do other clean ups and closures ...
	}

FUNCTION	**getVxiDataLibVerboseFlag**
PURPOSE	Allows the state of the verbose flag to be determined.
SYNTAX	`int getVxiDataLibVerboseFlag ( void );`
INCLUDES	`#include rdDataLib.h`
RETURN VALUES	Returns an integer value of 0 (ie. Zero or False or Off) when the verbose flag is false, otherwise a non-zero is returned indicating the flag is True or On.
DESCRIPTION	This function returns the state of the VXI data library global verbose text output flag.
USEAGE	#include "rdDataLib.h" ... Other includes ... printf ("Verbose Flag is %s\n", (getVxiDataLibVerboseFlag () ? "On" : "Off")); ... Other statements ...

FUNCTION	opnLSWT_VxiModuleDataBase	
PURPOSE	Initialises the VXI library code so that VXI data can be accessed on Bernoulli.	
SYNTAX	`int opnLSWT_VxiModuleDataBase ( int verboseFlag );`	
	`int verboseFlag;`	Used to set a global flag to indicate if verbose message output is required. If it is set to zero then error messages are the only text output from this and other VXI data access functions.
INCLUDES	`#include rdDataLib.h`	
RETURN VALUES	Returns an integer value of 0 (ie. Zero or False) if an error occurs, otherwise a non-zero is returned indicating a successful open operation.	
DESCRIPTION	This function is the VXI data access initialisation routine. It opens access to the shared memory containing data sent from the VXI module through HP Vee and the vxiDataD daemon. The code copies the "verboseFlag" parameter to a global flag variable then attaches the two VXI shared memory areas and the VXI semaphore to the program calling this open routine. If a shared memory cannot be attached or the semaphore is not available then an error message is output and the return value is set to zero. If the code is successful then it sets a global flag indicating that the other VXI module data routines can be used and returns a non-zero value.	

USEAGE	`#include "rdDataLib.h"`
	… Other includes …
	`int flag = 1;`
	… Do other initialisation …
	`if ( ! opnLSWT_VxiModuleDataBase( flag )) fprintf ( stderr, "Open Failed\n"`
	`);`
	`else {`
	    … Read data …
	`if ( ! clsLSWT_VxiModuleDataBase()) fprintf ( stderr, "Close Failed\n" );`
	    … Do other clean ups and closures …
	`}`	

FUNCTION	rdLSWT_VxiDataLinesProcessed	
PURPOSE	Allows data transfer rate checking.	
SYNTAX	`int rdLSWT_VxiDataLinesProcessed ( unsigned long int * countPtr)`	
	`Unsigned long int *countPtr;`	Pointer to an unsigned long integer into which to put the current number of data lines processed by the vxiDataD daemon. It must not be a NULL pointer.

INCLUDES	`#include rdDataLib.h`
RETURN VALUES	Returns an integer value of 0 (ie. Zero or False) if an error occurs, otherwise a non-zero is returned indicating a successful read operation.
DESCRIPTION	This function gives access to the VXI data line count. The data line count is incremented by one each time a complete line of numbers from HP Vee is converted to an array of double in shared memory. This count parallels the packet count in a UDP based data transfer system. This function first checks that there has been a successful open (see previously "opnLSWT_VxiModuleDataBase()") and if so it then copies the data line count from shared memory to the calling programs memory. If there has not been a previous successful open then an error message is output and the return value is set to zero. If the code is successful then it returns a non-zero value.

USEAGE	`#include "rdDataLib.h"`
	… Other includes …
	`unsigned long int cnt, oldCnt;`
	… initialize other variables …
	`if ( ! opnLSWT_VxiModuleDataBase( 0 )) fprintf ( stderr, "Open Failed\n" );`
	`else {`
	`  if ( ! rdLSWT_VxiDataLinesProcessed ( &cnt))`
	`    fprintf ( stderr, "Unable to Read Data Line Count\n" );`
	`  else {`
	`    oldCnt = cnt;`
	`    do { rdLSWT_VxiDataLinesProcessed ( &cnt ); } while ( cnt == oldCnt );`
	`        … Read up to date data …`
	`  }`
	`}`

FUNCTION	rdLSWT_VxiModuleData	
PURPOSE	Allow access to latest VXI data plus the time stamp and total number of data values associated with that latest data.	
SYNTAX	int rdLSWT_VxiModuleData (double * dataBffr, int * valuesInBffrPtr, int * totalNumberAvlblPtr, struct timeval * timeStmpPtr)	
	Double *dataBffr;	Pointer to a target buffer into which to put the double length floating point VXI data. This pointer must not be a NULL pointer.
	int *valuesInBffrPtr;	Pointer to an integer that is the desired number of VXI data values to copy on entry and on exit is the actual number of VXI data values in the data buffer. This pointer must not be a NULL pointer and the integer it points to on entry must be greater than zero.
	int *totalNumberAvlblPtr;	Pointer to an integer that is set to the total number of VXI data values in the shared memory data buffer on exit. This pointer may be a NULL pointer and if it is then this parameter is ignored.
	Struct timeval * timeStmpPtr;	Pointer to a target buffer into which to put the time stamp associated with the VXI data values. This pointer may be a NULL pointer and if it is then this parameter is ignored.
INCLUDES	#include lswtGlbl.h #include rdDataLib.h	
RETURN VALUES	Returns an integer value of 0 (ie. Zero or False) if an error occurs, otherwise a non-zero is returned indicating a successful read operation.	
DESCRIPTION	This function gives access to the VXI data plus additional information. It copies the latest available double length floating point data values plus time stamp and total number of available data values from the shared memory that is updated by the vxiDataD daemon. This function first checks that there has been a successful open (see previously "opnLSWT_VxiModuleDataBase()") and if so it copies the data and additional information from the shared memory to the calling programs memory. If there has not been a previous successful open then an error message is output and the	

	return value is set to zero. If the code is successful then it returns a non-zero value.

USEAGE	#include "rdDataLib.h"
	… Other includes …
	int totalCnt, dataCount = 1;
	double dataBuffer[4];
	struct timeval tmStamp;
	… Initialize other variables …
	if (! opnLSWT_VxiModuleDataBase(0)) fprintf (stderr, "Open Failed\n");
	else {
	  if (! rdLSWT_VxiModuleData (dataBuffer, &dataCnt, &totalCnt, &tmStamp))
	    fprintf (stderr, "Unable to Read Data\n");
	  else {
	    printf ("There are %d data values available, ", totalCnt);
	    printf ("the first is %lf\n", dataBuffer[0]);
	    printf ("The data was placed in shared mem at %d sec\n", tmStamp.tv_sec);
	  }
	}

FUNCTION	**rdLSWT_VxiModuleDoubles**	
PURPOSE	Allow access to latest VXI data.	
SYNTAX	`int rdLSWT_VxiModuleDoubles ( double * dataBffr, int * valuesInBffrPtr)`	
	`double *dataBffr;`	Pointer to a target buffer into which to put the double length floating point VXI data. It must not be a NULL pointer.
	`Int *valuesInBffrPtr;`	Pointer to an integer that is the desired number of VXI data values to copy on entry and on exit is the actual number of VXI data values in the data buffer. This pointer must not be a NULL pointer and the integer it points to on entry must be greater than zero.
INCLUDES	`#include rdDataLib.h`	
RETURN VALUES	Returns an integer value of 0 (ie. Zero or False) if an error occurs, otherwise a non-zero is returned indicating a successful read operation.	
DESCRIPTION	This function gives access to the VXI data. This function first checks that there has been a successful open (see previously `"opnLSWT_VxiModuleDataBase() "`) and if so it copies the latest available double length floating point data values from the shared memory to the calling programs memory. If there has not been a previous successful open then an error message is output and the return value is set to zero. If the code is successful then it returns a non-zero value.	

USEAGE	`#include "rdDataLib.h"`
	 ... Other includes ...
	 int cnt, dataCount = 64;
	 double dataBuffer[64];
	 ... initialize other variables ...
	 if (! opnLSWT_VxiModuleDataBase(0)) fprintf (stderr, "Open Failed\n");
	 else {
	   if (! rdLSWT_VxiModuleDoubles (dataBuffer, &dataCnt))
	    fprintf (stderr, "Unable to Read Data\n");
	   else {
	     for(cnt = 0; cnt < dataCount; cnt++) printf ("%lf\n", dataBuffer[cnt]);
	   }
	 }	

FUNCTION	setVxiDataLibVerboseFlag	
PURPOSE	Allows the state of the verbose flag to be changed at some point after initialisation of the library code.	
SYNTAX	`void setVxiDataLibVerboseFlag ( int flag );`	
	`int flag;`	Used to set the global verbose flag to indicate if verbose message output is required. If it is set to zero then error messages are the only text output from this and other VXI data access functions.
INCLUDES	`#include rdDataLib.h`	
RETURN VALUES	None.	
DESCRIPTION	This function sets the state of the VXI data library verbose text output flag. The code sets the global verbose text output flag equal to the result of comparing the flag parameter with false (ie. zero).	

USEAGE	#include "rdDataLib.h"
	… Other includes …
	int flag = 1;
	… Do other initialisation …
	if (! opnLSWT_VxiModuleDataBase(flag)) fprintf (stderr, "Open Failed\n");
	else {
	   flag = 0;
	   setVxiDataLibVerboseFlag(flag);
	   … Read Data (without any verbose messages) …
	}	

DISTRIBUTION LIST

Software Function Summaries for the Low Speed Wind Tunnel Data Acquisition System

Craig D. Edwards, Owen F. Holland, Stephen S. Lam
Sunny Y. F. Leung, Yoel Y. Link and Robert Toffoletto

AUSTRALIA

DEFENCE ORGANISATION

Task Sponsor

S&T Program
    Chief Defence Scientist
    FAS Science Policy   } shared copy
    AS Science Corporate Management
    Director General Science Policy Development
    Counsellor Defence Science, London (Doc Data Sheet only)
    Counsellor Defence Science, Washington (Doc Data Sheet only)
    Scientific Adviser to MRDC Thailand (Doc Data Sheet only)
    Scientific Adviser Policy and Command
    Navy Scientific Adviser (Doc Data Sheet and distribution list only)
    Scientific Adviser - Army (Doc Data Sheet and distribution list only)
    Air Force Scientific Adviser
    Director Trials

    Aeronautical and Maritime Research Laboratory
    Director

    Chief of Air Operations Division
    N. Pollock
    N. Matheson
    Y. Link
    C. Edwards (2)
    O. Holland
    S. Lam
    R. Toffoletto
    P. Malone
    H. Quick

DSTO Library and Archives
    Library Fishermans Bend (Doc Data Sheet)
    Library Maribyrnong (Doc Data Sheet)
    Library Salisbury (1 copy)
    Australian Archives
    Library, MOD, Pyrmont (Doc Data sheet only
    US Defense Technical Information Center, 2 copies
    UK Defence Research Information Centre, 2 copies
    Canada Defence Scientific Information Service, 1 copy

NZ Defence Information Centre, 1 copy
National Library of Australia, 1 copy

Capability Systems Staff
Director General Maritime Development (Doc Data Sheet only)
Director General Aerospace Development (Doc Data Sheet only)
Director General C31 Development (Doc Data Sheet only)

Army
SO (Science), DJFHQ(L), MILPO Enoggera, Queensland 4051 (Doc Data Sheet only)

Intelligence Program
DGSTA Defence Intelligence Organisation
Manager Information Centre, Defence Intelligence Organisation

Corporate Support Program
Library Manager, DLS-Canberra (Doc Data Sheet only)

UNIVERSITIES AND COLLEGES
Australian Defence Force Academy
    Library
    Head of Aerospace and Mechanical Engineering
Hargrave Library, Monash University (Doc Data Sheet only)
Librarian, Flinders University

OTHER ORGANISATIONS
NASA (Canberra)
AusInfo

<p align="center">OUTSIDE AUSTRALIA</p>

ABSTRACTING AND INFORMATION ORGANISATIONS
Library, Chemical Abstracts Reference Service
Engineering Societies Library, US
Materials Information, Cambridge Scientific Abstracts, US
Documents Librarian, The Center for Research Libraries, US

INFORMATION EXCHANGE AGREEMENT PARTNERS
Acquisitions Unit, Science Reference and Information Service, UK
Library - Exchange Desk, National Institute of Standards and Technology, US
SPARES (5 copies)

Total number of copies: **44**

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA	1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE Software Function Summaries for the Low Speed Wind Tunnel Data Acquisition System	3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION) Document (U) Title (U) Abstract (U)

4. AUTHOR(S) Craig D. Edwards, Owen F. Holland, Stephen S. W. Lam, Sunny Y. F. Leung, Yoel Y. Link and Robert Toffoletto	5. CORPORATE AUTHOR Aeronautical and Maritime Research Laboratory PO Box 4331 Melbourne Vic 3001 Australia

6a. DSTO NUMBER DSTO-TN-0321	6b. AR NUMBER AR-011-643	6c. TYPE OF REPORT Technical Note	7. DOCUMENT DATE November 2000

8. FILE NUMBER M1/8/1348	9. TASK NUMBER RDI 98/179	10. TASK SPONSOR DST	11. NO. OF PAGES 213	12. NO. OF REFERENCES 7

13. URL ON THE WORLD WIDE WEB http://www.dsto.defence.gov.au/corporate/reports/ DSTO-TN-0321.pdf	14. RELEASE AUTHORITY Chief, Air Operations Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, SALISBURY, SA 5108

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CASUAL ANNOUNCEMENT	Yes

18. DEFTEST DESCRIPTORS

low speed wind tunnels, data acquisition, graphical user interfaces (computer systems), computer programs

19. ABSTRACT

The data acquisition system in the Low Speed Wind Tunnel at the Aeronautical and Maritime Research Laboratory was recently upgraded. Several software packages were developed using C, X/Motif, and OpenGL programming languages and libraries. The software provides wind tunnel operators with graphical user interfaces from which test data can be easily monitored, acquired and processed. Descriptions of all software source code functions have been compiled and included as Appendices in this document. These function summaries have also been published on the Defence Science and Technology Organisation Intranet for easy reference for future software development.